VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2017 Jakub Přibyl

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

ONLINE DETEKTOR BODŮ ZÁJMU
ONLINE INTEREST POINT DETECTOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Jakub Přibyl**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Jan Mašek, Ph.D.**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

Student: Jakub Přibyl ***ID:*** 174384
Ročník: 3 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Online detektor bodů zájmu

POKYNY PRO VYPRACOVÁNÍ:

Prozkoumejte možnosti trénováni online detektoru pro sledování objektů. Zprovozněte tento detektor a proveďte jeho vylepšení podle zadání vedoucího práce. Dále natrénujte nové modely, u kterých ověřte přesnost detekce na vybraných příkladech a výsledky vykreslete do grafů.

DOPORUČENÁ LITERATURA:

[1] Z. Kalal, K. Mikolajczyk, and J. Matas, “Tracking-Learning-Detection,” Pattern Analysis and Machine Intelligence 2011

[2] Z. Kalal, K. Mikolajczyk, and J. Matas, “Face-TLD: Tracking-Learning-Detection Applied to Faces,” International Conference on Image Processing, 2010.

Termín zadání: 1.2.2017 ***Termín odevzdání:*** 8.6.2017

Vedoucí práce: Ing. Jan Mašek, Ph.D.
Konzultant:

doc. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Tato práce se věnuje problematice online učení detektoru při dlouhodobém sledování objektu ve videosekvenci. Tento objekt je definován pomocí ohraničujícího obdelníku. V práci jsou popsány jednotlivé části detektoru: sledování objektu, detekce objektu a online učení detektoru. Hlavním přínosem práce je rozšíření programu OpenTLD o paralelní detekci a sledování více objektů současně. Paralelizace je pak porovnána na několika praktických příkladech a je porovnán vliv procesoru při detekci. Nejlepších výsledků bylo dosaženo při paralelizaci s detekováním všech objektů. Nejpřesnější detekce byla v případě dostatečně naučených objektů při nejmenší změně podoby.

KLÍČOVÁ SLOVA

Detekce objektu, OpenCV, online učení detektoru, paralelizace, sledování objektu, TLD

ABSTRACT

This thesis focuses on online learning detector for long-term tracking of object in video sequence. The object is defined by a bounding box. The text describes different parts of the detector: object tracking, object detection and online learning detector. The main contribution of this work is creating extension of the OpenTLD program for parallel detection and tracking of multiple objects. The parallelization is then compared on two practical examples and the processor's impact on detection is compared. The best results were achieved with parallelization, where all objects were detected. The most accurate detection was in the case of sufficiently learned objects with the smallest shape change.

KEYWORDS

Object detection, OpenCV, online learning detector, parallelization, object tracking, TLD

PŘIBYL, Jakub *Online detektor bodů zájmu*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2017. 52 s. Vedoucí práce byl Ing. Jan Mašek, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Online detektor bodů zájmu“ jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Janu Maškovi Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.
Dále bych chtěl poděkovat mé matce za dlouhodobou podporu při studiu.

Brno ....

podpis autora(-ky)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Počítačové vidění je velmi aktuální a rozšířené odvětví výpočetní techniky, které se zabývá softwarem na získávání informací ze zachyceného obrazu. Počítačové vidění se například používá k ovládání procesů, detekce jevů, modelování objektů a sledování objektů. Tato práce je zaměřena především na sledování objektů s online učením, které může být rozděleno do tří části: sledování objektu, detekce objektu a učení detekce.

Cílem bakalářské práce je prozkoumat možnosti trénování online detektoru pro sledování objektu, zprovoznění takovéhoto detektoru, natrénování nových modelů a rozšíření programu o sledování více objektů současně. Sledovač, který je využit v této práci se nazývá OpenTLD a jeho autorem je Georg Nebehay [1]. Tento sledovač využívá knihovny OpenCV [2] k detekci objektu a používá se k dlouhodobému sledování objektu. OpenTLD využívá přístupu tzv. rekurzivního sledování, které je založeno na odhadu optického toku pomocí metody od Lucase a Kanadeho [3]. Pro detekci objektu se získávají šablony pomocí P-N učení. Pro porovnávání šablon se využívá kaskádové architektury, která se skládá ze tří částí: filtr odchylky, souborový klasifikátor a klasifikátor k-nejbližších sousedů.

V rámci této práce byl zprovozněn OpenTLD detektor ke sledování objektu. K tomu byly použity dostupné zdrojové kódy, které byly s pomocí programu CMake, prostředí Visual Studia a knihovny OpenCV, použity pro vytvoření spustitelné aplikace schopné sledovat objekt zájmu. Byl vytvořen i návod na zprovoznění tohoto detektoru. Dále byly popsány jednotlivé části detektoru a probrána problematika sledování, učení a detekce a ověřena jejich funkčnost.

Hlavním přínosem práce je rozšíření programu OpenTLD o paralelní detekci pomocí OpenMP a sledování více objektů najednou. Bylo porovnáno několik různých vstupů, na kterých byla ověřena správnost detekce a výkon při paralelizaci. Nejlepších výsledků bylo dosaženo při paralelizaci, kdy všechny objekty byly detekovány. Nejlepší schopnost detekce měly dostatečné naučené objekty s minimální změnou tvaru. Naopak nejhorších výsledků bylo dosaženo při málo naučených objektech, které rychle měnili svůj tvar.

V kapitole 1 je popsáno krátkodobé a dlouhodobé sledování a k čemu se dnes oba tyto způsoby využívají. V 2. kapitole je rozebrána problematika učícího se detektoru s rekurzivním sledováním a jejich jednotlivými složkami: sledování objektu, detekce objektu a online učení. V kapitole 3 je uvedena knihovna OpenCV, program pro sestavení projektu CMake a postup instalace detektoru OpenTLD. Ve 4. kapitole jsou popsány problémy související se sledováním objektu, natrénované modely a jejich funkčnost. Dále je v kapitole popsána paralelizace, rozšíření pro sledování více objektů a ukázky praktických příkladů.

1 ONLINE DETEKCE, SLEDOVÁNÍ A UČENÍ V SOUČASNOSTI

Online detektor bodů zájmu ve videosekvenci je dlouhodobě vyvíjená problematika. Sledování a detekce je používano v mnoha odvětvích, především pro sledovací kamery, ale třeba i k vědeckému průzkumu (např. ke sledování zvěře a analýze jeho chování [4]). Velkým problémem je především měnící se velikost objektu, rotace, vzhled, popřípadě zmizení objektu ze záběru. V podkapitolách bude popsáno fungování krátkodobého sledování a jakým způsobem se přistupuje k dlouhodobému sledování.

1.1 Krátkodobé sledování

Krátkodobé sledování funguje tak, že si předává informace mezi jednotlivými snímky a nepočítá s tím, že by se objekt mohl vytratit z obrazu. Tyto metody sledování se zaměřují spíše na rychlost a přesnost, než na dlouhodobou spolehlivost. Pro zlepšení jejich dlouhodobé spolehlivosti, je možné je určitým způsobem zdokonalit, jakmile ale dojde k selhání detekce, samotný detektor pak není schopen se obnovit.

První metoda, kterou lze krátkodobě sledovat se nazývá rekurzivní sledování. Ta porovnává předešlí snímek s aktuálním a provádí odhad, kde by se daný objekt mohl nacházet. Objekt musí být po celou dobu viditelný, pokud přestane, nastává problém, protože detekce není schopna se obnovit. Klasickým příkladem je sledovací algoritmus pojmenovaný podle jeho tvůrcu Lucase a Kanadeho [3, 5].

Dalším způsobem sledování je pomocí detekce (sledování s využitím šablon [6]). Využívá data, které má detektor předem připravené a natrénované. Objekt, který chceme sledovat se potom porovnává s těmito daty a zjišťuje se mezi nimi shoda. Problém nastává pokud objekt změní razantně svoji podobu, která potom není totožná s daty v databázi. Zároveň nemůžeme sledovat objekt, o kterém předem nemáme nějaké informace.

V dnešní době se spíše používají tzv. sledovače s učením, díky kterým je možnost objekt sledovat po delší dobu i přes určitě změny vzhledu nebo zmizení ze záběru. V článku [7] se autoři zabývají používáním rekurzivní metody k nalezení deformované tkáně pomocí informací, které jsou typické pro tuto tkáň (textura, barva). V článku [8] je rozebráno, jakým způsobem by se dalo zakomponovat učení do sledování s využitím šablon. V práci [9] Amanda Berg popisuje, jak funguje detekování u infra červených kamer a vysvětluje jejich problematiku. Vyvrací v práci tvrzení, jak podobné si jsou metody detekování pro normální kamery a infra červené kamery.

1.2 Dlouhodobé sledování objektu pomocí učení detektoru

Dlouhodobé v tomto případě znamená, že videosekvence může být nekonečně dlouhá, obsahuje různé střihy, rychlé pohyby kamerou a sledovaný objekt se může i dočasně vytratit ze záběru. Proto je potřeba u takovýchto sledovačů, aby byly schopny znovu detekovat objekt i po tom, co se vrátil po dlouhé době do záběru. Různým změnám vzhledu se věnují takzvané adaptivní sledovací metody [10]. Tyto metody jsou dvě, podle toho jakým způsobem se aktualizují.

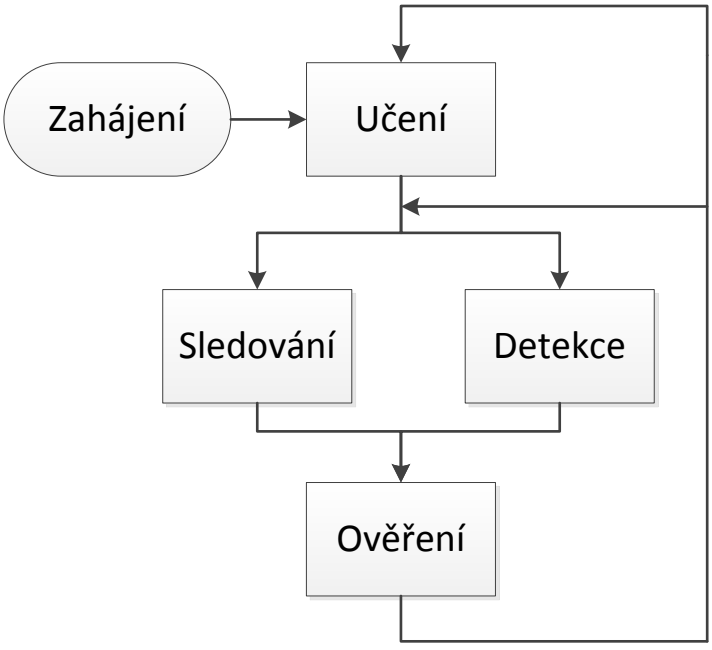
První z nich je aktualizace každého snímku [11], která je běžná pro adaptivní sledovače. Tento způsob je rychlejší, protože se předpokládá, že sledovač pracuje správně a na tomto základě aktualizuje objekt. Tento fakt znamená, že sledovač je schopen se rychle přizpůsobit vzhledu objektu, ale to taky může vést k tomu, že sledovač začne selhávat.

Druhou metodou je takzvané učení s dozorem (semi-supervised learning [12]). Zde se dělí data na dva druhy: označená a neoznačená, kde označená data jsou v menším zastoupení než neoznačená. Díky kombinaci poskytují značné zlepšení v přesnosti učení. Označená data jsou data, která byla pořízená před detekcí a neoznačená jsou data, která jsou získávána při detekci a chodu programu. Tato metoda sledování je efektivní v mnoha odvětvích a je v poslední době asi nejvíce vyvíjena. Například v článku[13] je řešeno třídění obrázků do různých kategorií, pomocí více pohledů a učení s dozorem. V trénovací fázi je potřeba získat data na pár cvičných snímcích a pomocí více pohledů, které využívají ze začátku označená data, je zajištěna potřebná efektivita.

Ne vždy jsou k dispozici snímky, které by byly v dobré kvalitě a tak občas můžou vznikat problémy. S tímto se snaží vypořádat v práci [14] Jiří Matas, který byl i u zrodu TLD (tracking - learning - detection), o kterém si povíme v další kapitole. Testovali znovu detekovatelnost objektu a drift i u videosekvence, která měla skoro 30 000 snímku. Tato práce zajišťuje o něco lepší výsledky i u méně kvalitních snímků, díky takzvaným virtuálním rohům. V [15] autoři popisují využití dlouhodobého sledování v reálném čase na po zemi se pohybujících objektech pro bezpilotní letadla, která používají *Camshift* algoritmus. Ten je založen na barvách a rysech cílového objektu. Má dobrý výkon a je robustní i přes to, že cílový objekt je občas natočen nebo lehce deformován.

2 UČÍCÍ SE DETEKTOR

Tato kapitola popisuje učící se detektor, který slouží ke dlouhodobému sledování. Objekt, který bude sledován je označen v jediném snímku, ten se však nemusí objevovat v každém záběru, ale může i ze záběru zmizet. Největším problémem je zajistit sledování i potom, co se do záběru vrátí, jelikož může změnit svůj vzhled. Pro úspěšné dlouhodobé sledování je nutné, aby detektor zvládl změny tvarů objektu, změny prostředí a operoval v reálném čase. Tímto se zabývá framework s názvem TLD[1] [16], který dlouhodobé sledování rozděluje do tří částí, které pracují zároveň: sledování, učení a detekce. Sledovač sleduje objekt na každém snímku. Detektor lokalizuje všechny objekty, které byly doposud spatřeny a pokud je to nutné, opravuje sledovač. Učení zjišťuje chyby detektoru a aktualizuje ho, aby předešel chybám v budoucnu. Proces sledování je inicializován zvolením objektu zájmu. Žádné další zásahy od uživatele nejsou nutné.



Obr. 2.1: Vývojový diagram průběhu TLD.

TLD využívá metodu rekurzivního sledování, konkrétněji metodu od Lucase a Kanadeho [3], která vypočítává odhad optického toku pro sadu bodů v ohraničujícím obdelníku. Tento ohraničující obdelník je zvolen uživatelem v prvním snímku

[1] `http://personal.ee.surrey.ac.uk/Personal/Z.Kalal/tld.html`

videosekvence okolo objektu zájmu. Spolu s touto metodou se využívají ke sledování chyb další dvě: metoda křížové korelace a dopředně-zpětné chyby. Pomocí toho, jsou vyfiltrovány body, které se mezi dvěma snímky nezměnili a pomocí nich se určí ohraničující obdelník dalšího snímku.

Detektor je založen na metodě klouzavého okna a kaskádového klasifikátoru. Metoda klouzavého okna postupně skenuje celý snímek a vytvoří mnoho obrazových částí, které potom procházejí jednotlivými fázemi kaskádového klasifikátoru [17]: filtr odchylky, souborový klasifikátor a klasifikátor nejbližších sousedů. Jednotlivé fáze buď propouštějí nebo ve většině případů zahazují tyto obrazové části. Pokud je obrazová část propuštěna všemi třemi fázemi je vyhodnocena jako objekt zájmu.

P-N učení se stará o dlouhodobé sledování objektu v reálném čase. V každém snímku se vyhodnocuje detektor, jeho chyby a pomocí těchto informací se aktualizuje a předchází problémům v budoucnu. Dělá se to pomocí dvou expertů: P-expert rozeznává přehlédnuté detekce a N-expert rozpoznává falešné poplachy. P- N experti dělají chyby samy o sobě, ale jsou na sobě nezávislé a tím pádem se můžou navzájem vykompenzovat.

Obr. 2.2: Označený objekt ke sledování. Obsah obdelníku je zvětšen vpravo.

V podkapitole 2.1 je rozebráno sledování objektu, metoda Lucase a Kanadeho a výpočty odhadu optického toku, chyb a transformačního modelu. Vysvětlen je i integrátor, který sdružuje ohraničující obdelníky sledovače a detektoru. V podkapitole 2.2 je popsána detekce objektu pomocí metody klouzavého okna a jednotlivé filtrování obrazových částí pomocí kaskádové architektury. V podkapitole 2.3 jsou vysvětleny jednotlivé metody učení a je také popsáno P-N učení, které využívá právě TLD.

2.1 Sledování objektu

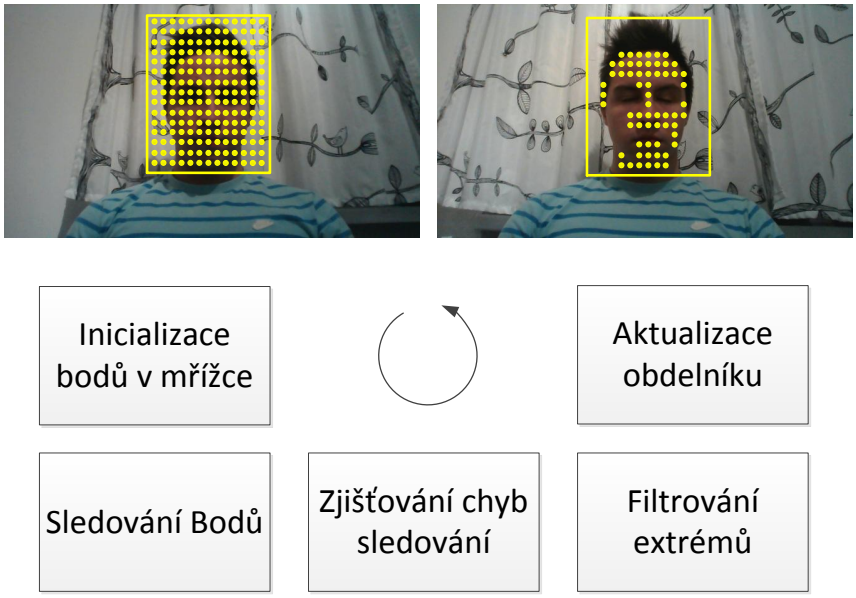
Sledování má za úkol předpovídat pohyb objektu. Většinou se předpokládá, že je celou videosekvenci viditelný. Tento objekt je popsán pomocí šablony a jeho pohyb je předpovídán díky minimálnímu rozdílu mezi jednotlivými snímky. Tento způsob sledování může být realizován buď staticky (šablona se nemění) nebo se přizpůsobuje (šablona je extrahována z minulého snímku). Šablona může být obrázek nebo barevný histogram, který je uložený v tréninkové sadě nebo v případě TLD je vybrán v prvním snímku. Pokud se vzhled objektu příliš neliší od šablony, pak se všechny parametry ze šablony ve sledování přiřadí objektu zájmu. Šablony mají limitovanou modelovou přizpůsobivost, protože zastupují pouze jediný vzhled objektu.

Pokud je však změna větší, už vzniká problém. Proto existují sledovače, které dokáží modelovat kromě samotného objektu i okolní prostředí, a tak eliminovat nežádoucí jevy. Toto generování může probíhat buď *offline* nebo v reálném čase. Prostředí je generováno jako negativní třída, kterou musí sledovač rozlišovat. Lze k tomu přistupovat tak, že si sledovač vytvoří hranici, která rozlišuje mezi objektem a okolním prostředím. Statické rozlišovací sledovače trénují objekt před samotným sledováním, což limituje jeho aplikaci u známých objektů, přizpůsobující se rozlišovací sledovač to dělá při sledování.

Hlavní částí je *update*, který okolí objektu rozděluje podle vzdálenosti. Blízké okolí se přiřazuje ke vzorkům pozitivního trénování a vzdálené okolí ke vzorkům negativního trénování, ty se potom používají k třídění v každém snímku. Tato strategie zvládá velké rozdíly objektu mezi jednotlivými snímky, ale problém nastává v momentě, kdy objekt není ve videu delší dobu.

2.1.1 Sledování v implementaci OpenTLD

OpenTLD využívá rekurzivní metodu sledování. Před samotným sledováním je nutno zkonstruovat ohraničující obdelník kolem objektu, který chceme sledovat. V tomto obdelníku se nachází určitá sada bodů, které mezi sebou mají mezeru. Potom je odhadován optický tok pro každý tento bod pomocí metody od Lucase a Kanadeho [3]. Tato metoda funguje nejvíce spolehlivě pro body, které jsou v rozích obdelníku a není schopna sledovat body, které se nachází v homogenních oblastech. OpenTLD využívá informace jak z metody od Lucase a Kanadeho, ale také ze dvou různých typů sledování chyb založené na křížové korelaci a dopředně-zpětné chyby, aby se vyfiltrovali sledovací body, které jsou nejpravděpodobněji chybné. Na obrázku 2.3 vpravo jsou ukázány zbylé body. Tyto zbylé body pak slouží k předpovědi pozice nového ohraničujícího obdelníku v dalším snímku pomocí transformačního modelu, který je založen na změnách tvaru a velikosti objektu.



Obr. 2.3: Princip rekurzivního sledování.

2.1.2 Odhad optického toku metodou od Lucase a Kanadeho

Lucasova a Kanadeho metoda je založena na třech faktorech. Prvním z nich je takzvaná stálost jasu [18]. Ta je vyjádřena jako:

$$I(X) = J(X + d). \tag{2.1}$$

Vzorec 2.1 poukazuje na to, že pixel ve 2D soustavě (X) ve snímku I může změnit pozici v dalším snímku J, avšak jeho jas zůstane stejný. Proměnná d je označována jako vektor přemístění. Dalším faktorem se nazývá časové přetrvání. To je vyjádřeno jako:

$$J(X) \approx I(X) + I'(X)d, \tag{2.2}$$

kde d je vyjádřeno:

$$d \approx \frac{J(X) - I(X)}{I'(X)}, \tag{2.3}$$

pro jakýkoliv pixel. Hodnota vektoru d je malá. $I'(X)$ je v tomto případě gradient snímku I ve 2D soustavě X. Třetím faktorem je prostorová koherence. Ta deklaruje, že všechny pixely v okně okolo určitého pixelu se pohybují koherentně. Pomocí této úvahy je d zjištěno díky minimalizaci nejmenších čtverců

$$\sum_{(x,y)\in W} \left(J(X) - I(X) - I'(X)d\right)^2, \tag{2.4}$$

kde W je definováno jako oblast okolo každého pixelu.

2.1.3 Výpočet chyb

Kalal společně s kolegy navrhli dopředně-zpětné chybové měření [19]. Tato metoda je zobrazena na obrázku 2.4. Zde je vidět, že bod 1 je na obou snímcích, proto je tento bod stabilní a sledovač je schopen ho vysledovat na stejném místě. U bodu dva je však problém ten, že na dalším snímku už tento bod není a proto sledovač není schopen ho najít. Tento způsob měření je definován jako Euklidova vzdálenost:

$$\varepsilon = |p - p''|, \tag{2.5}$$

kde p'' je:

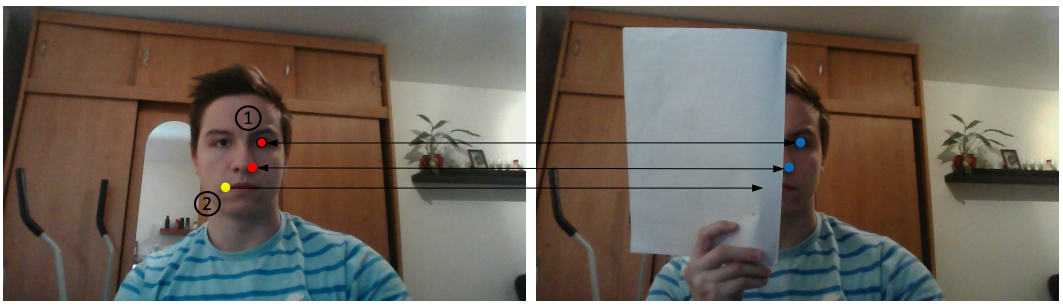
$$p'' = LK(LK(p)). \tag{2.6}$$

LK v tomto případě znamená Lukasova a Kanadeho metoda a je aplikována dvakrát na p.

Dopředně zpětná metoda je ještě použita ve spojení s dalším měření chyb, založené na podobnosti obklopujících oblastí p a výsledek sledovače pro obklopující oblasti p'. Podobnost oblastí P_1 a P_2 je porovnána pomocí Normalizovaného korelačního koeficientu NCC:

$$NCC(P_1, P_2) = \frac{1}{n-1} \sum_{x=1}^{n} \frac{(P_1(x) - \mu_1)(P_2(x) - \mu_2)}{\sigma_1 \sigma_2}, \tag{2.7}$$

kde μ_1, μ_2, σ_1, σ_2 jsou střední a směrodatné odchylky P_1 a P_2. Normalizovaný korelační koeficient je neměnný oproti jednotným změnám jasu.



Obr. 2.4: Dopředně-zpětné měření chyb.

2.1.4 Transformační model

S využitím přístupu z [19], se vypočítává medián všech dopředně-zpětných chyb med_{FB} a medián všech měření podobnosti med_{NCC}. Zachovají se pouze ty body dopředně-zpětných chyb, které jsou menší než med_{FB} a body měření podobnosti větší než med_{NCC}. Pokud je med_{FB} větší než předefinovaný práh θ_{FB}, nevrací se žádné výsledky a měření se považuje za nespolehlivé. Zbývající body jsou použity ke kalkulaci transformace ohraničujícího obdelníku. Vzdálenosti mezi všemi páry bodů jsou vypočteny před a po sledování. Relativní nárůst je interpretován změnou v měřítku. Osa x je vypočítána pomocí mediánu horizontálního posunu všech bodů. Pro osu y platí to stejné, akorát je vypočítán medián vertikálního posunu bodů.

Algoritmus 1: Rekurzivní sledování

Input: B_I: ohraničující obdelník, I: původní snímek, J: další snímek

forall p_i **do**

- $p'_i \leftarrow LK(p_i)$;
- $p''_i \leftarrow LK(p'_i)$;
- $\varepsilon_i \leftarrow |(p_i) - (p''_i)|$;
- $\eta_i \leftarrow NCC(W(p_i),W(p'_i))$;

end

$med_{NCC} \leftarrow \mathrm{median}(\eta_1 ... \eta_2)$;

$med_{FB} \leftarrow \mathrm{median}(\varepsilon_1 ... \varepsilon_2)$;

if $med_{FB} > \theta_{FB}$ **then**

- $B_J = \emptyset$;

else

- $C \leftarrow (p_i,p'_i) \mid p'_i \neq \emptyset,\ \varepsilon_i \leq med_{FB},\ \eta_i \geq med_{NCC}$;
- $B_J \leftarrow \mathrm{transform}(B_I,C)$;

end

Output: B_J: další ohraničující obdelník

2.1.5 Integrátor

Integrátor sdružuje ohraničující obdelník sledovače a ohraničující obdelník detektoru do jednoho výstupní ohraničujícího obdelníku TLD. Pokud ohraničující obdelník není stanoven sledovačem ani detektorem, objekt je označen jako neviditelný. V opačném případě se získá z integrátoru maximálně jistý ohraničující obdelník, změřený pomocí konzervativní podobnosti S^c. Sledovač a detektor mají identické priority, ale zastupují zásadně rozdílné odhady stavu objektu. Mezitím, co detektor vyhledává již známe šablony, sledovač vyhledává možné nové šablony a tím pádem získává nová data pro detektor.

2.2 Detekce objektu

Detekce objektu má za úkol lokalizovat objekt ve snímku. Metody detekování jsou založeny buď na funkcích snímku nebo na metodě klouzavého okna. Přístup k detekci, který je založen na funkcích snímku většinou následuje kostru: Funkce detekce, funkce rozpoznání a zařazení modelu. Ve většině případů je využívána rovinnost nebo celý 3D model. Tyto algoritmy dosáhly takové úrovně, že můžou v reálném čase fungovat i na slabších zařízeních a zvládnout detekovat i velký počet objektů. Omezením je však to, že je dopředu potřeba znát geometrie objektu.

Metody, které jsou založeny na klouzavém okně spočívají v tom, že vstupní snímek oskenují pomocí okna, které má neurčitou velikost a pro každé okno se musí rozhodnout, zda podokna obsahují objekt zájmu nebo ne. Aby se dosáhlo provedení v reálném čase, tak si detektory založené na klouzavém okně adoptovaly takzvanou kaskádovou architekturu [17]. Díky faktu, že pozadí je mnohem více frekventované než objekt zájmu, je klasifikátor rozdělen do několika fází. Trénování takovéhoto detektoru vyžaduje vysoký počet trénovacích příkladů a náročné výpočty, aby bylo možné rozeznat hranici mezi objektem a pozadím.

V TLD se používá metoda klouzavého okna, která rozhoduje v každé obrazové části, zda-li obsahuje objekt. Generovány jsou takové obrazové části, které mají jinou velikost a tvar než původní ohraničující obdelník. Po určitých krocích změny, je schopno se vytvořit okolo 50 000 pod-oken pro snímek o rozlišení QVGA (240x320). Konkrétní číslo záleží na poměru stran počátečního ohraničujícího obdelníku. Dále používá kaskádový klasifikátor, který je rozdělen do tří fází: filtr odchylky, souborový klasifikátor a klasifikátor nejbližších sousedů. Každá z těchto fází buď odmítne podokno nebo je pošle do další části.

2.2.1 Metoda klouzavého okna

V klouzavém okně pro detekci objektu se postupným posouváním ohraničujícího obdelníku po snímku vyhledává objekt zájmu. Tato metoda však předpokládá to, že každá tato oblast ho může obsahovat. Díky tomuto však vzniká obrovský počet pod-oken, které ovlivňují výkon systému. Proto se používá pár předpokladů, které zredukují počet teoretických oblastí objektu. Každá vygenerovaná část obrazu projde kaskádovou architekturou, kde je buď snímek zahozen pomocí jednotlivých částí architektury nebo projde a bude vyhodnocen jako detekovaný objekt viz. 2.5.

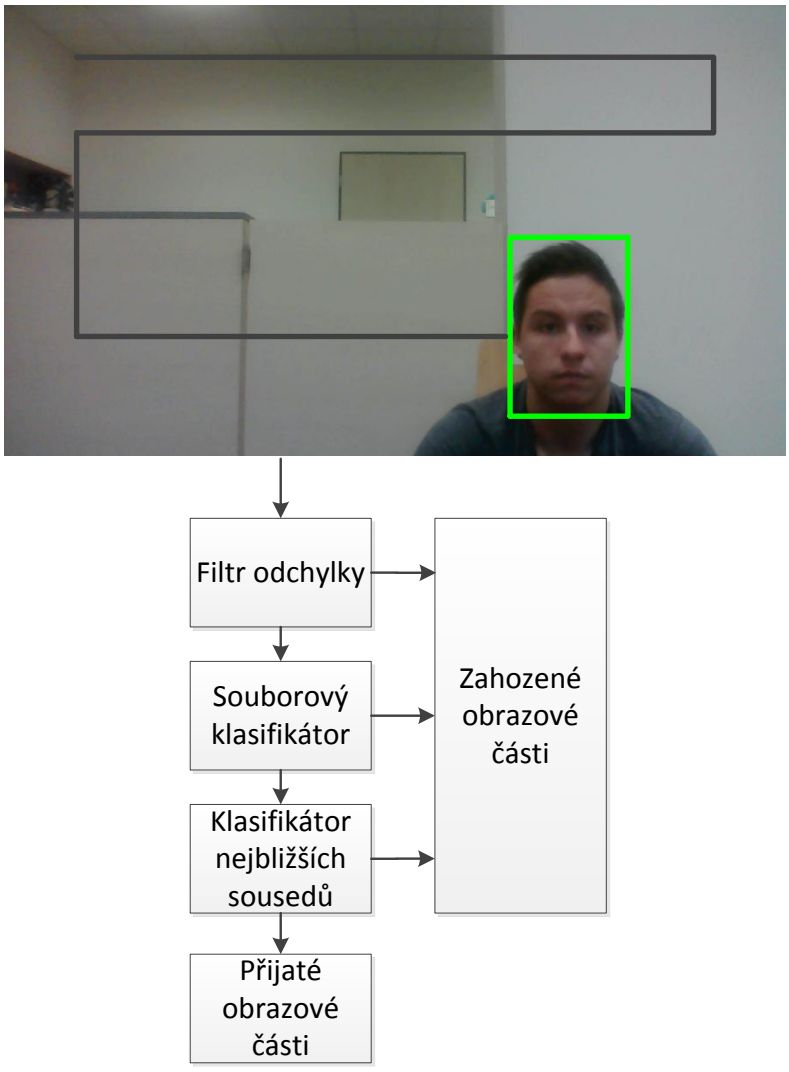
Jedním z těchto předpokladů je ten, že objekt zájmu si zachovává stejný poměr stran. Klouzavé okno pak mění svoji velikost pomocí několika parametrů během pár kroků. V [16] upravují velikost pomocí následujících parametrů: kroková změna měřítka = 1,2, horizontální krok = 10 % šířky, vertikální krok = 10 % výšky. Minimální

velikost pod-oken je 20 pixelů.

V [1] je to mírně poupravené tím, že minimální velikost pod-oken je 25 pixelů. Dále vypočítávají velikost množiny všech pod-oken R zredukované omezením, které je:

$$|R| = \sum_{s \in 1,2^{-10,10}} \left\lceil \frac{n - s(w + d_x)}{s d_x} \right\rceil \left\lceil \frac{m - s(h + d_x)}{s d_y} \right\rceil, \tag{2.8}$$

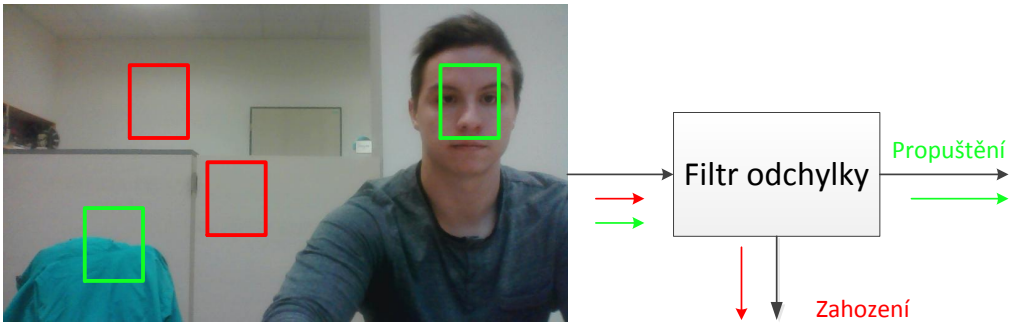
kde w je šířka pod-okna, h je výška pod-okna, s je kroková změna měřítka, d_x a d_y jsou kroky nastavení horizontální a vertikální osy.



Obr. 2.5: Ve snímku se používá metoda klouzavého okna.

2.2.2 Filtr odchylky

Filtr odchylky je první fází kaskádové architektury. Tato část zamítá veškeré části obrazu, pro které je odchylka šedé menší než 50 % té části, která byla zvolena ke sledování. Na snímku 2.6 jsou tyto části označené červeně. Tato fáze využívá toho, že hodnota odchylky šedé části p může být vyjádřena jako $\mathrm{E}(p^2) - \mathrm{E}^2(p)$ a očekávaná hodnota $\mathrm{E}(p)$ může být změřena v konstantním čase pomocí integrálních obrazů [17]. Tento filtr ve většině případů odmítá víc než 50 % částí, které neobsahují objekt (například obloha, ulice). Práh odchylky omezuje maximální změnu objektu.



Obr. 2.6: Filtr odchylky. Zelené oblasti jsou propuštění do souborového klasifikátoru.

2.2.3 Souborový klasifikátor

Souborový klasifikátor je druhou fází detektoru. Vstupními daty jsou obrazové části, které nebyly odmítnuty filtrem odchylky. Soubor se skládá z n jednoduchých klasifikátorů. Každý jednoduchý klasifikátor i provádí několik porovnání pixelů v obrazové části, z čehož vzniká binární kód x, který indexuje pole posteriorních pravděpodobností $P_i(y|x)$, kde $y \in \{0,1\}$. Pravděpodobnosti jednotlivých jednoduchých klasifikátorů jsou zprůměrovány a soubor klasifikuje obrazovou část jako objekt, jestliže je průměrná pravděpodobnost větší než 50%.

Porovnání pixelů

Každý jednoduchý klasifikátor je založen na sadě porovnání pixelů. Toto porovnání je generováno náhodně *offline* a zůstává neměnné v reálném čase. V prvním kroku je snímek rozmazán se směrodatnou odchylkou 3 pixelů ke zvýšení odolnosti proti změnám a obrazovému šumu. V dalším kroku je předefinovaná sada porovnání pixelů natažena do obrazových části. Každé toto porovnání vrací buď 0 nebo 1 a tyto měření jsou seřazení do řetězce x.

Generování porovnání pixelů

Důležitým elementem souborového klasifikátoru je nezávislost jednoduchých klasifikátorů. Nezávislost klasifikátoru je podporována generováním rozdílných pixelových porovnávání pro každý jednoduchý klasifikátor. Prvně je omezen prostor pro pixely v obrazové části a vygenerovány jsou všechny možné horizontální a vertikální pixelové porovnání. Potom je zaměněno pořadí jednotlivých porovnání a rozděleno do jednoduchých klasifikátorů. Výsledkem by mělo být, že každý klasifikátor je založen na odlišných sadách funkcí a všechny funkce dohromady by měly pokrýt celou obrazovou část.

Posteriorní pravděpodobnost

Každý jednoduchý klasifikátor i udržuje rozložení posteriorních pravděpodobností $P_i(y|x)$ [20]. Toto rozložení má 2^d vstupů, kde d je počet porovnání pixelů. TLD využívá 13 porovnání, které dávají 8192 možných kódů a ty indexují posteriorní pravděpodobnost. Tato pravděpodobnost je dána jako

$$P_i\left(y|x\right) = \frac{p}{p+n}, \tag{2.9}$$

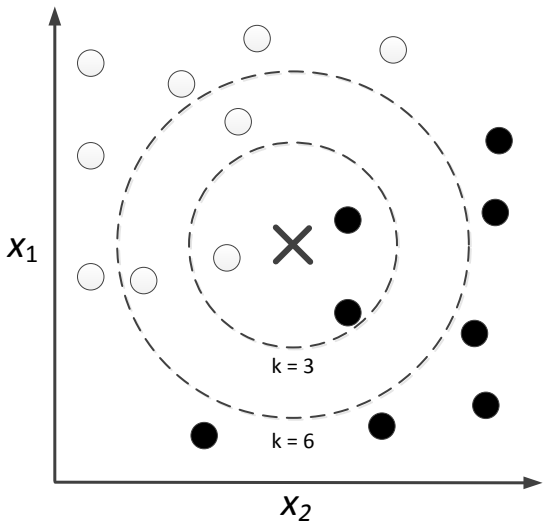
kde p a n jsou pozitivní a negativní obrazové části, kterým byl přiřazen stejný binární kód.

Spuštění a update

Ve spouštěcí části, všechny základní posteriorní pravděpodobnosti jsou nastaveny na nulu. Během reálného času je souborový klasifikátor aktualizován. Označené příklady jsou tříděny pomocí souboru a jestli je třídění nesprávné, potom jsou aktualizovány odpovídající p a x a zároveň aktualizují i $P_i(y|x)$.

2.2.4 Klasifikátor *k*-nejbližších sousedů

Obecný princip je takový, že po roztřídění obrazových částí se do této fáze dostane asi jen okolo 50 snímků. V tomto případě se tedy používá online model, který roztřídí obrazové části pomocí k-NN klasifikátoru [21]. k-NN je ve zkratce klasifikátor *nearest-neighbor* neboli klasifikátor nejbližších sousedů. Výkon k-NN klasifikátoru závisí na výpočtu vzdálenosti mezi obrazovými částmi, které se dostaly až do této fáze. Obrazové části se porovnávají se šablonami uložených vzorů a vstupních oblastí, kde se porovnává pixel po pixelu. Pozitivně klasifikované obrazové části reprezentují odpovědi detektoru objektů. Pokud počet šablon v k-NN klasifikátoru překročí určitou hranici, která je určena pamětí, tak se použije náhodné zapomínání šablon. Kalal a spol. objevili, že počet šablon se stabilizuje okolo několika stovek a ty můžou být lehce uloženy v paměti. Na obrázku 2.7 jde vidět, jakým způsobem k-NN funguje. V tomto případě se využívá tréninková sada, která se skládá ze dvou tříd (A a B) s několika instancemi, které jsou označeny bílými a černými tečkami. Dále se využívají tzv. atributy (x_1 a x_2) k rozlišení mezi těmito třídami ve 2D prostředí. Prostředí může být až nD, kde počet atributů je roven počtu dimenzí prostředí. Je potřeba přiřadit neoznačené pozorování (křížek) k jedné z těchto tříd. Toho je docíleno vzdáleností mezi tímto neoznačeným pozorováním a jednotlivými instancemi. Pro případ, kdy $k = 3$, jsou dva sousedé černé barvy a jeden bílé. To znamená, že toto pozorování bude označeno jako člen třídy B. Pokud bude $k = 6$, potom bude neoznačené pozorování označeno jako člen třídy A, protože v této oblasti jsou tři instance třídy A a pouze dvě instance třídy B.



Obr. 2.7: Dvě třídy (A a B) jsou rozlišeny tečkami. A - bílé tečky, B - černé tečky.

2.3 Učení

Detektory objektů jsou běžně trénovány za předpokladu, že všechny tréninkové vzory jsou označeny. U TLD je ale potřeba, aby detektor byl trénován jen pomocí jediného vzoru a videa. Tento problém se dá formulovat jako učení s dohledem (semi-supervised) [12, 22], který pracuje s daty označenými a neoznačenými. Tyto metody většinou pracuji s nezávislými a identicky distribuovanými daty s určitými vlastnostmi, jako třeba, že neoznačené příklady formují „přirozené“ klastry ve vymezeném prostoru. Řada algoritmů se spoléhá na podobné předpoklady, které byly navrženy v minulosti, včetně očekávání maximalizace, sebe-učení, kooperativní učení.

2.3.1 Metody učení s dohledem

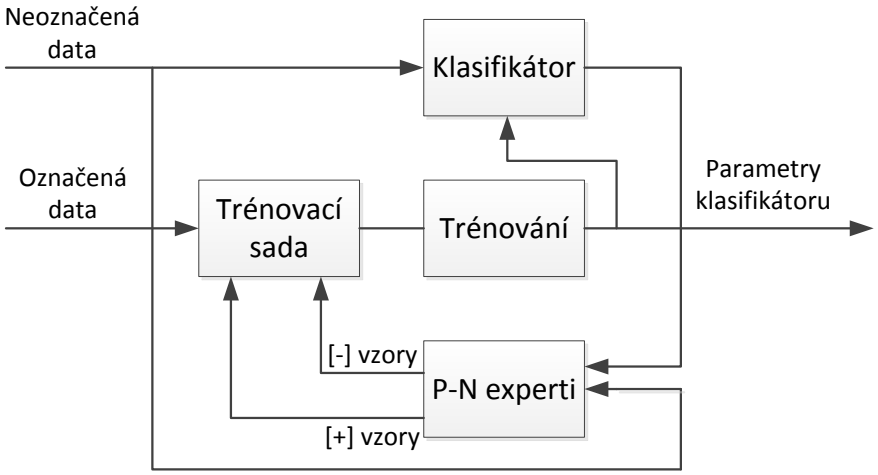
Očekávání maximalizace (EM - Expectation-Maximization) je generická metoda pro hledání odhadu parametrů modelu, které jsou předány pomocí neoznačených dat. EM je iterativní proces, který v případě binární klasifikace zaměňuje informační štítky neoznačených dat a trénuje klasifikátor. Tyto informační štítky reprezentují buď numerickou hodnotu pravděpodobnosti, např. na kolik procent má pacient určitou nemoc nebo kvalitativní kategorie, která je vyjádřena například slabým nebo silným souhlasem s tím, že pacient určitou nemoc má. V učení s dohledem se EM algoritmus spoléhá na předpoklady s tzv. separací s nízkou hustotou [23]. Tato metoda je použita například ke klasifikaci dokumentů.

Sebe-učení začíná trénováním původního klasifikátoru z označené tréninkové sady, klasifikátor je potom vyhodnocen v neoznačených datech. Vzory s největší jistotou odpovědi klasifikátoru jsou přidány do tréninkové sady a klasifikátor je zachován. Bylo ale zjištěno, že detektor zlepšil více svůj výkon, pokud neoznačená data byla zvolena spíše nezávislým měřením než jistotou klasifikátoru. Bylo doporučeno, že separace s nízkou hustotou je pro tento způsob nedostatečná a jiné metody by mohli fungovat lépe. Tato metoda je použita třeba k detekci lidského oka [24].

Kooperativní učení [25] je metoda učení založená na nezávislých klasifikátorech, které mohou trénovat jeden druhého. Aby se vytvořil takový nezávislý klasifikátor, kooperativní trénování předpokládá, že jsou k dispozici dvě nezávislé oblasti. Učení je započato trénováním dvou oddělených klasifikátorů pomocí označených vzorů. Oba klasifikátory jsou potom vyhodnoceny jak neoznačená data. S jistotou označené vzorky z prvního klasifikátoru jsou použity pro zlepšení tréninkové sady druhého klasifikátoru a naopak v iterativním procesu. Metoda je použita třeba v dopravě nebo k detekci pohybujícího se objektu.

2.3.2 P-N Učení

P-N učení je jednou z hlavních částí TLD frameworku. Cílem tohoto učení je vylepšit výkon detektoru díky online zpracování videa. V každém snímku videa je potřeba vyhodnotit současný detektor, identifikovat jeho chyby a aktualizovat je, aby se předešlo případným chybám v budoucnu. Hlavní myšlenkou P-N učení je, že chyby detektoru můžou být identifikovány jako dva druhy „expertů“. **P-expert** identifikuje jenom falešně negativní a **N-expert** identifikuje jenom falešně pozitivní. Obě tyto části dělají samy o sobě chyby, ale jejich nezávislost na sobě umožňuje kompenzaci těchto chyb.



Obr. 2.8: Blokový diagram P-N učení.

P-N učení se skládá ze čtyř bloků: klasifikátor k výuce, trénovací sada, trénování s dohledem - metoda která trénuje klasifikátor pomocí trénovací sady a P-N experti, kteří generuje pozitivní nebo negativní tréninkové vzory během učení viz. 2.8.

Tréninkový proces je zahájen vložením označené sady $L = \{(x,y)\}$ z trénovací sady. Trénovací sada je potom předána do učení s dohledem, kde se trénuje klasifikátor neboli zjistí se počáteční parametry Θ^0. Proces učení poté pokračuje iterativně. V Iteraci k, je klasifikátor, který byl vytrénován v předchozí iteraci a označuje celou neoznačenou sadu, $y_u^k = f(x_u|\Theta^{k-1})$ pro všechny $x_u \in X_u$, kde X_u je neoznačená sada, x_u je vzor z vyznačené oblasti a f je klasifikátor. Klasifikace je analyzována pomocí P-N expertů, kteří zjistí, jaké vzory byly klasifikovány špatně. Tyto vzory jsou přidány do trénovací sady se změněným označením. Iterace se dokončí po přetrénování klasifikátoru zjištěním parametrů Θ^k.

Důležitým elementem P-N učení je zjištění chyb klasifikátoru. Hlavní myšlenkou je oddělit zjištěné falešně negativní od falešné pozitivních. Díky tomuto faktu je neoznačená sada rozdělena do dvou částí podle momentálního označení a každá část je analyzována nezávislým expertem. P-expert analyzuje vzory označené jako negativní, odhaduje falešné negativní a přidává je do tréninkové sady s pozitivním označením. V iterace k je výstup P-experta $n^+(k)$ pozitivních vzorů. N-expert analyzuje vzory označené jako pozitivní, odhaduje falešně pozitivní a přidává je do tréninkové sady s negativním označením. V iteraci k je výstup N-experta $n^-(k)$ negativních vzorů. P-expert zvyšuje obecnost klasifikátoru a N-expert zvyšuje rozlišnost klasifikátoru.

Za předpokladu, že označené sady X_u jsou známé, je jednoduché rozeznat chybně označené vzory a přidat je do tréninkové sady se správným označením. Takováto strategie se běžně nazývá svépomocná [26]. Klasifikátor trénovaný svépomocně s dohledem se zaměřuje na hranice rozhodnutí a většinou překoná klasifikátor, který byl trénovaný pomocí náhodné vzorové tréninkové sady. Stejný nápad se zaměřením na hranice rozhodnutí tvoří základy P-N učení s tím rozdílem, že označená sada X_u je neznámá. Na P- N učení se lze tím pádem podívat jako na zobecnění standardního samozaváděcího programu v případě, kde označení nejsou dána, ale spíše přiřazena díky P-N expertům. Jako každý jiný proces i P-N experti dělají chyby v označeních.

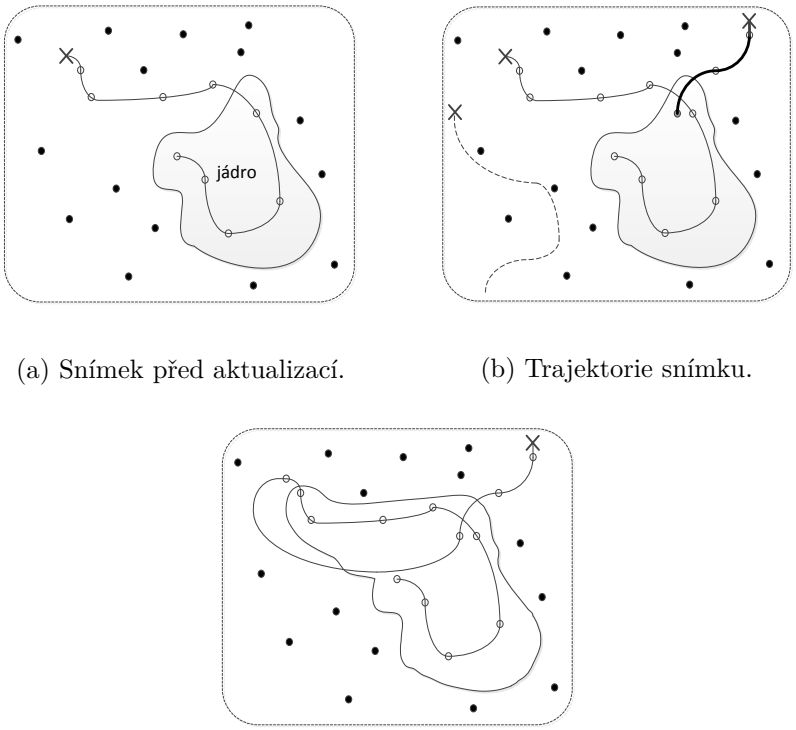
2.3.3 P-expert

Cílem P-expertu je objevování nových vzhledů objektu a tím pádem zvýšit zobecnění detektoru objektů. V TLD systému je dráha objektu generována kombinací sledovače, detektoru a integrátoru (viz. 2.1.5). Tento kombinovaný proces sleduje přerušovanou trajektorii, která není vždy správná. Úkolem P-expertu je v tomto případě identifikovat spolehlivé části trajektorie a použít ke generaci pozitivních tréninkových vzorů.

Aby se identifikovaly spolehlivé části trajektorie, tak se P-expert spoléhá na model objektu M. Model objektu je reprezentován bílými body ve vyznačeném místě viz 2.9. Kladné vzory jsou bílé tečky spojené křivkou, která udává jejich pořadí, záporné vzory jsou černé tečky. Pomocí konzervativní podobnosti S^c může být definována podmnožina ve vyznačeném prostoru, kde S^c je větší než hranice. Tato podmnožina se nazývá jádro modelu objektu. Je třeba si uvědomit, že jádro není statická struktura, ale roste s tím, jak se dostávají nové vzory do modelu. Tento růst je ale pomalejší než celý model.

P-expert identifikuje spolehlivé části trajektorie tímto způsobem. Trajektorie se stane spolehlivou hned po vstupu do jádra a zůstane spolehlivá, dokud není znova zahájena nebo sledovač rozpozná svoji chybu. Obrázek 2.9 je rozdělen na tři části:

a) Model objektu a jádro ve vyznačeném prostředí, b) Nespolehlivá (čárkovaná) a spolehlivá (vyznačená silnou čarou) trajektorie, c) Model objektu a jádro po aktualizaci. Bílé tečky jsou kladné vzory, černé jsou záporné a křížek znamená konec trajektorie. V každém snímku P-expert rozhoduje o spolehlivosti aktuálního umístění. Pokud je aktuální umístění spolehlivý, potom P-expert generuje sadu kladných vzorů, které aktualizují model objektu a souborový klasifikátor.



(a) Snímek před aktualizací.

(b) Trajektorie snímku.

(c) Snímek po aktualizaci.

Obr. 2.9: Princip P-expertu.

2.3.4 N-expert

N-expert generuje negativní tréninkové vzory. Jeho úkolem je objevit zaplnění v pozadí. Základním předpokladem N-expertu je, že objekt může zabírat maximálně jedno místo ve snímku. Pokud je tedy známa poloha objektu, okolí této polohy je označeno jako negativní. N-expert je aplikován ve stejném čase jako P-expert, pokud je trajektorie spolehlivá. V tomto případě, obrazové části, které jsou vzdáleny od současného ohraničujícího obdelníku (překrytí $< 0{,}2$) jsou označeny jako negativní. Pro aktualizaci detektoru a souborového klasifikátoru se musí počítat jen s těmi obrazovými částmi, které nebyli odmítnuty filtrem odchylky ani souborovým klasifikátorem.

3 NÁVRH ŘEŠENÍ A POSTUP INSTALACE

Jedním z cílů práce je zprovoznit detektor pro sledování objektů. Pro sledování objektů je v této práci využito knihoven OpenTLD a OpenCV. OpenTLD je C++ implementace TLD frameworku, kterou vytvořil G. Nebehay v rámci své diplomové práce [1]. OpenCV se využívá pro zpracování obrazu a u OpenTLD je využita pro detekrovou část. OpenTLD vyžaduje pro sledování vytvořit ohraničující obdelník okolo objektu, který chceme sledovat. To je provedeno buď v prvním snímku nebo po zmáčknutí klávesové zkratky. Poté je objekt sledován a dále zpracováván. OpenTLD je schopno si brát data buď z kamerového výstupu, sekvence obrázků nebo z videa, ale pouze videa s koncovkou `.avi` Vygenerování projektu a propojení OpenCV a OpenTLD je vyřešeno pomocí programu CMake. Sestavení projektu se potom dělá pomocí Visual studia.

V následující podkapitole 3.1 je popsáno, co to vlastně je OpenCV, k čemu se využívá, z čeho se skládá a jakou hraje roli v knihovně OpenTLD. V podkapitole 3.2 je vysvětleno, k čemu slouží CMake. V poslední podkapitole 3.3 je vysvětlena instalace OpenTLD a sestavení projektu pro správně spuštění.

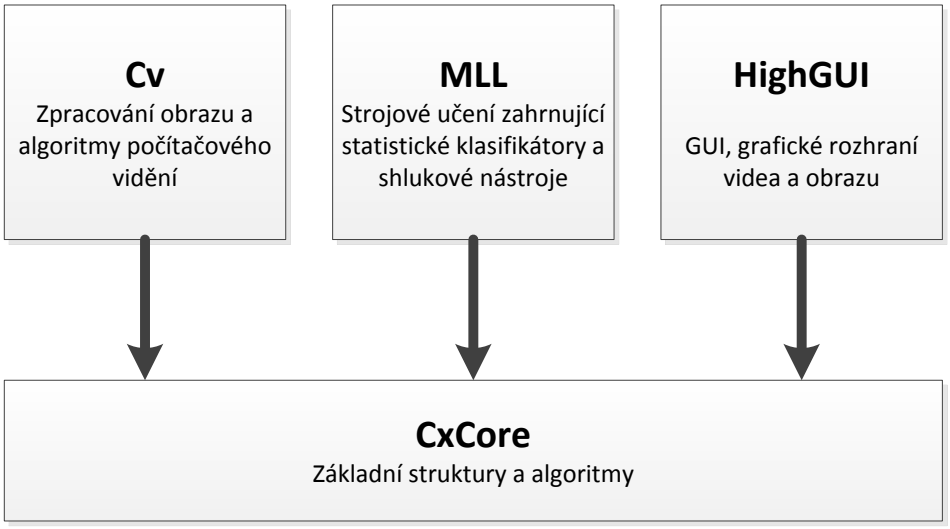
3.1 OpenCV - prostředí pro práci s obrazem

OpenCV je open source knihovna pod volnou licencí, která se využívá ke zpracování obrazu. Je licensovaná pod BSD, což znamená, že je možné knihovnu i její obsah svobodně šířit a povoluje její modifikaci. Tato knihovna obsahuje přes 2500 optimalizovaných algoritmů, které se dají použít například k rozeznání obličejů, detekce objektů, sledování pohybu na kamerách, extrahování 3D modelů nebo mazání červených očí z fotek pořízených s bleskem. OpenCV má rozhraní pro C++, C, Python, Java nebo Matlab a podporuje operační systémy Windows, Linux, Android a Mac OS. Samotná knihovna je napsaná v C++.

Struktura OpenCV se skládá ze čtyř hlavních částí:

- **cxcore**
  Zajišťuje sadu základních datových struktur. To zahrnuje např. obrazy, matice, body. Zároveň také obsahuje základní kreslící funkce, které zvládají kreslit čáry, kruhy atd. v obraze.
- **highgui**
  Poskytuje jednoduché možnosti uživatelského rozhraní, několik kodeků pro obraz a videa, možnosti nahrávání obrazu a podobně. Pro pokročilejší využití UI se musí použít frameworku jako Qt, Winforms.

- **cv**
  Zahrnuje funkce pro zpracování obrazu a počítačové vidění (např. funkce pro detekci hrany, kalibrace kamery, optický tok).
- **mll**
  Strojové učení je sada tříd a funkcí pro statistické klasifikace, regrese a slučování dat.



Obr. 3.1: Struktura knihoven OpenCV.

OpenTLD využívá z knihovny OpenCV tři moduly: cxcore, highgui a cv. Díky modulu cv je zajištěna detekce objektu. Je důležitou částí knihoven OpenCV a je významnou částí použita v kaskádové architektuře OpenTLD. Pomocí highgui je vyrešena problematika základního uživatelského rozhraní, kde je potřeba zobrazit uživateli nějakým způsobem buď kamerový výstup nebo video. Základní funkce jsou pak zajištěny pomocí cxcore.

3.2 CMake

CMake v této práci hraje významnou roli, protože díky němu je možné vygenerovat projekt OpenTLD a sestavit ho tak, aby byl funkční. Díky němu je provedeno i spojení knihoven OpenCV a OpenTLD. CMake je rozšířitelný, open-source systém, který řídí proces sestavení v operačním systému a kompilátoru nezávislým způsobem. Na rozdíl od mnoha multiplatformních systémů, CMake je určen pro použití

s nativním prostředím. Jednoduchý konfigurační soubor `CMakeLists.txt`, který je vložen ve složce, kde se nachází zdrojové kódy, je použit ke generování projektů. Tyto soubory jsou například makefiles na Unix nebo projekty na Windows. CMake dokáže vytvořit nativní prostředí, které dokáže zkompilovat zdrojový kód, vytvářet knihovny a budovat spouštěcí soubory. CMake také podporuje statické i dynamické knihovny. Příjemnou vlastností je i to, že vytváří soubor mezipaměti, který se používá s grafickým prostředím. CMake vyhledá spustitelné soubory, knihovny a pomocí grafického prostředí je možné si zvolit, jakým způsobem budou soubory sestaveny. CMake byl navržen hlavně pro podporu komplexních hiearchií adresářů a aplikací závislých na několika knihovnách.

3.3 Postup instalace

OpenTLD je možné nainstalovat na Windows, ale i na Linux, kde je postup odlišný. Tento návod je popsán pro instalaci na operační systém Windows. K instalaci byly použity tyto programy:

- OpenCV verze 2.4.XX [1]
- CMake verze 3.X.X [2]
- OpenTLD [3]
- Visual Studio 2015

Po stáhnutí všech programů je potřeba si na nějaké místo na disku rozbalit OpenCV a nainstalovat CMake. Prvním krokem je pomocí CMake sestavit projekt OpenCV, který by se dal zkompilovat. V CMake, po zvolení složky, kde se nachází `CMakeLists.txt` (`opencv/sources`) a složky pro sestavení OpenCV, je potřeba nastavit konfiguraci generování projektu. Pro tento návod je generován projekt pro Visual Studio 2015 Win64. Pro rychlejší a bezproblémové generování projektu je nutné vypnout tyto parametry:

- `BUILD/BUILD_DOCS`
- `BUILD/BUILD_PERF_TESTS`
- `BUILD/BUILD_TESTS`

Poté stlačit ještě jednou tlačítko Configure a pokud se nestane žádná chyba, tak stlačit tlačítko Generate. Po vygenerování projektu OpenCV je potřeba ho sestavit

[1] `http://opencv.org/downloads.html`
[2] `https://cmake.org/download/`
[3] `https://github.com/gnebehay/OpenTLD`

ve Visual Studiu v módu Release. Po sestavení projektu je potřeba ho „doinstalovat“ pomocí sestavení INSTALL. Tím se vytvoří složka install, kterou potřebujeme pro sestavení OpenTLD. Dále je potřeba nastavit v proměnném prostředí operačního systému cestu ke složce `bin`, která se nachází v `install/x64/vc14/bin`. Obdobným způsobem jako u OpenCV sestavíme projekt OpenTLD. Po vybrání složek a konfiguraci po nás CMake vyžaduje složku OpenCV. Je potřeba nastavit:

- `OpenCV/OpenCV_DIR` - cesta ke složce `install` v sestaveném projektu

Tímto krokem, by už CMake neměl vykazovat chyby a projekt by měl jít vygenerovat. Už stačí jen sestavit OpenTLD ve Visual Studiu v módu Release a program by se měl nacházet ve složce `bin`.

4 VÝSLEDKY PRÁCE

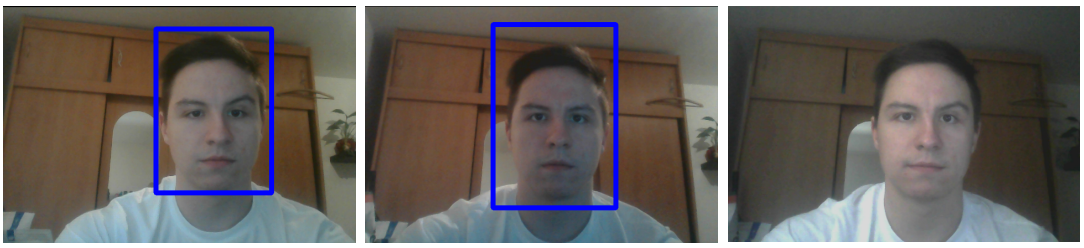
Cílem práce bylo zprovoznit detektor a natrénovat nové modely. Dalším rozšířením pak bylo zprovoznit paralelizaci a sledování více objektů současně. Jako detektor byl vybrán OpenTLD, u kterého je možné natrénovat modely přímo v programu a uložit je do textového souboru. Při trénování nových modelů bylo zjištěno, že existuje několik faktorů, které ovlivňují přesnost sledování, výkon a počet snímků za sekundu. Například při trénování pomocí webkamery bylo zjištěno, že podle prostředí se může měnit počet snímků za sekundu. Při sledování objektu ve videu platí totéž. Tato kapitola je rozdělena do čtyř částí, kde podkapitola 4.1 popisuje problémy se změnami objektu, podkapitola 4.2 uvádí natrénované modely. V podkapitole 4.3 je popsáno rozšíření pro sledování více objektů a v 4.4 jsou porovnány konkrétní příklady a zprovoznění paralelizace.

4.1 Změny u sledovaného objektu

U sledováného objektu se může vyskytnou několik změn, které mohou ovlivnit sledování nebo ho dokonce přerušit. Tyto změny mohou vzniknout u samotného objektu nebo díky vnějším vlivům. Nejčastější z těchto změn jsou: změna světla, částečné zakrytí objektu, rotace objektu, velikost objektu a nebo zmizení objektu ze záběru. Tyto změny byly otestovány v OpenTLD a je popsán jejich vliv na sledování objektu níže.

Změna světla v obraze

Při počátečním snímku 4.1a byly zapnuty světla na popředí a pozadí. Po menší změně (vypnutí světla na popředí) pokračovalo sledování objektu bez jakýchkoliv potíží viz 4.1b. Po vypnutí světla na pozadí však došlo k přerušení sledování viz 4.1c. Okamžitá velká změna světla v obraze je nežádoucí jev a je vhodné se této změně vyvarovat.

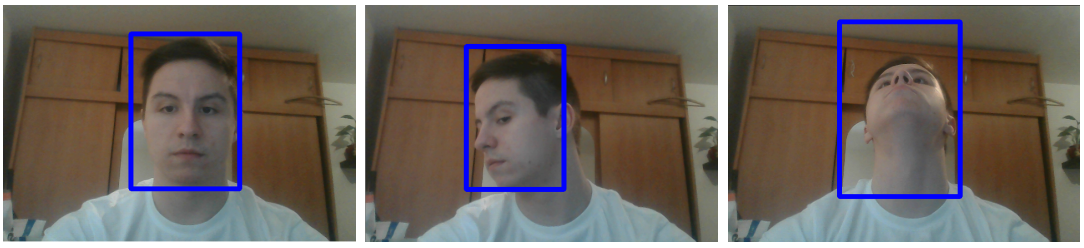


(a) Počáteční snímek (b) Malá změna světla (c) Velká změna světla

Obr. 4.1: Změna světla v obraze.

Rotace objektu

V této části bylo vyzkoušeno, jak sledování ovlivní rotace objektu do stran. Detektor u rotace neměl problém se sledováním objektu a pokud se objekt nějak razantně nebo rychle nezmění, tak se sledování nepřeruší.

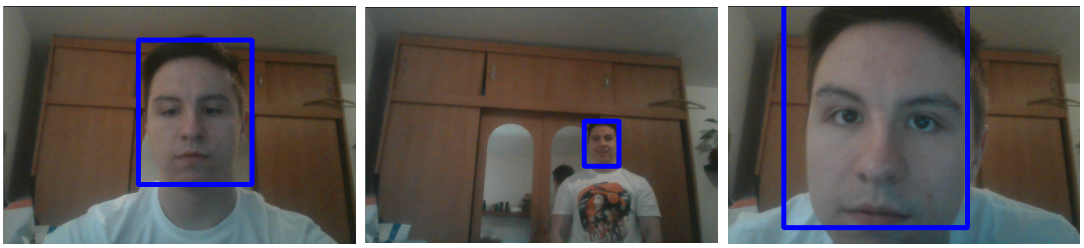


(a) Počáteční snímek (b) Rotace do strany (c) Rotace nahoru

Obr. 4.2: Rotace objektu

Velikost objektu

Další změnou objektu může být jeho velikost. Změna velikosti objektu je jedna z nejpravděpodobnějších věcí, které se můžou sledovanému objektu stát. OpenTLD si s tímto poradilo bez problémů a při zvětšení i zmenšení nedošlo k přerušení sledování. K přerušení by došlo v případě, že by objekt nebyl již v obraze.

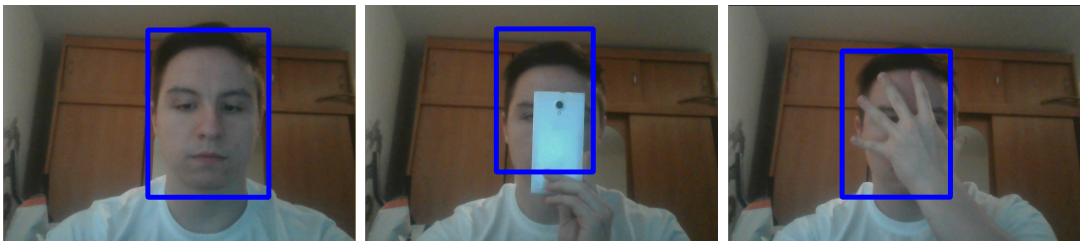


(a) Počáteční snímek (b) Malá velikost objektu (c) Velká velikost objektu

Obr. 4.3: Velikost objektu

Částečné zakrytí objektu

Občas se může sledovaný objekt dostat do situace, kdy je nějakým způsobem zakrytý. Přerušení sledování se pak odvíjí od toho, jakým způsobem je sledovaný objekt zakrytý. Pokud je zakrytý celý, tak se sledování přeruší. Na obrázku 4.4b a 4.4c jsou ukázány příklady, které sledování objektu nepřeruší.

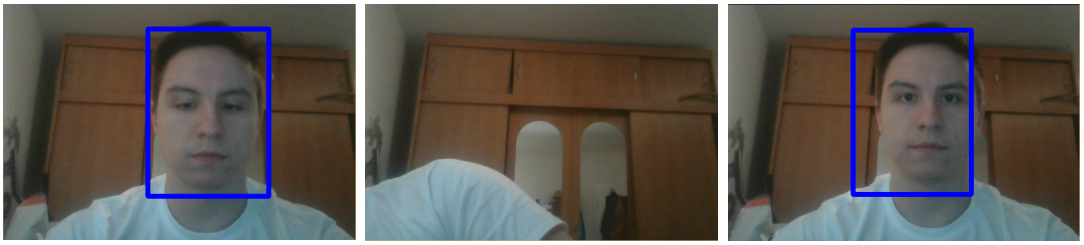


(a) Počáteční snímek (b) Zakrytí telefonem (c) Zakrytí rukou

Obr. 4.4: Částečné zakrytí sledovaného objektu

Vytracení objektu ze záběru

U dlouhodobého sledování je hlavním cílem obnovit sledování potom, co se objekt vytratí ze záběru. Na snímku 4.5a je vyznačen objekt, který je sledován. Na dalším snímku 4.5b objekt zmizí ze záběru a je přerušeno sledování objektu. Na posledním snímku 4.5c je obnoveno sledování po návratu objektu na webkameru.



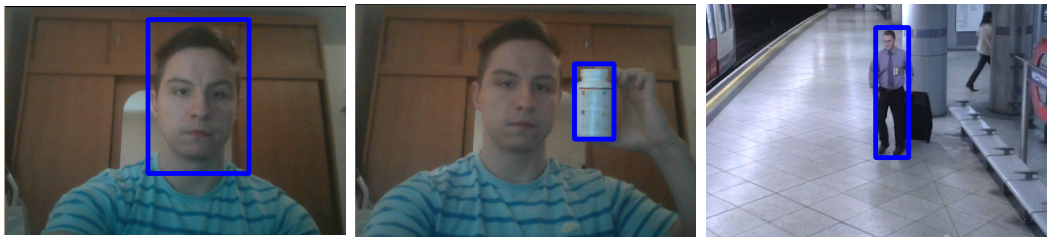
(a) Počáteční snímek (b) Objekt zmizel ze záběru (c) Objekt se vrátil do záběru

Obr. 4.5: Obnovení sledování objektu.

4.2 Trénování modelů

Byly trénovány tři modely: obličeje (obr. 4.6a), dózy (obr. 4.6b) a člověka (obr. 4.6c). Modely obličeje i dózy byly natrénovány pomocí webkamery a model člověka s pomocí video souboru. U modelu obličeje bylo vytvořeno 63 062 pod-oken, u modelu dózy bylo vytvořeno 60 620 a u modelu člověka 37 028. Při sledování ve videích mělo OpenTLD menší výkonnost, díky větším změnám v prostředí ohraničujícího obdelníku. Po opětovném spuštění a naimportování modelů fungovaly obličej i dóza bez problémů. U trénování člověka z videa nastal problém ve chvíli, kdy odcházel ze záběru. Ohraničující obdelník se zasekl uprostřed snímku a po návratu člověka do záběru nebyl schopen se zpátky chytit. Tento problém se vyskytoval i po více pokusech detekování stejného objektu. Po opětovném naimportování modelu se ohrani-

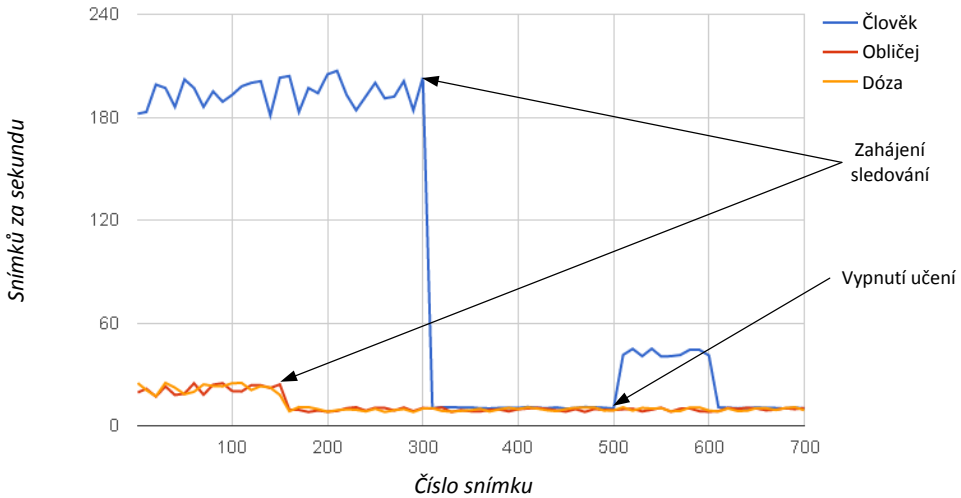
čující obdelník vůbec neobjevil. Video bylo získáno z datasetu AVSS [1]. Celkově bylo vyzkoušeno více videí, ale z důvodů velkého poklesu snímku za sekundu u některých videosekvencí a ve většině případech špatného sledování je OpenTLD vhodnější spíše pro použití webkamery, kde měl mnohem lepší výsledky a kde je možnost více natrénovat objekty.



(a) Model obličeje (b) Model dózy (c) Model člověka

Obr. 4.6: Ukázky trénovaných modelů

V grafu 4.7 je vidět, jaký je počet snímků za sekundu u každého modelu. U modelu obličeje a dózy byla použita webkamera, proto je počet snímků za sekundu mnohem menší než u modelu člověka. Před začátkem sledování se počet snímků za sekundu pohyboval okolo 20. Po vytvoření ohraničujícího obdelníku se snížila tato hodnota na 10 snímků za sekundu. Při spuštění videosekvence se před sledováním pohybovala hodnota snímků za sekundu mezi hodnotami 170 až 210. Po započetí sledování se tato hodnota snížila na 10. Dále je vidět, jakým způsobem ovlivnilo výkonnost vypnutí učení. U webkamery to nemělo prakticky žádný dopad, ale u videosekvence se počet snímků za sekundu zvýšil ke 40.



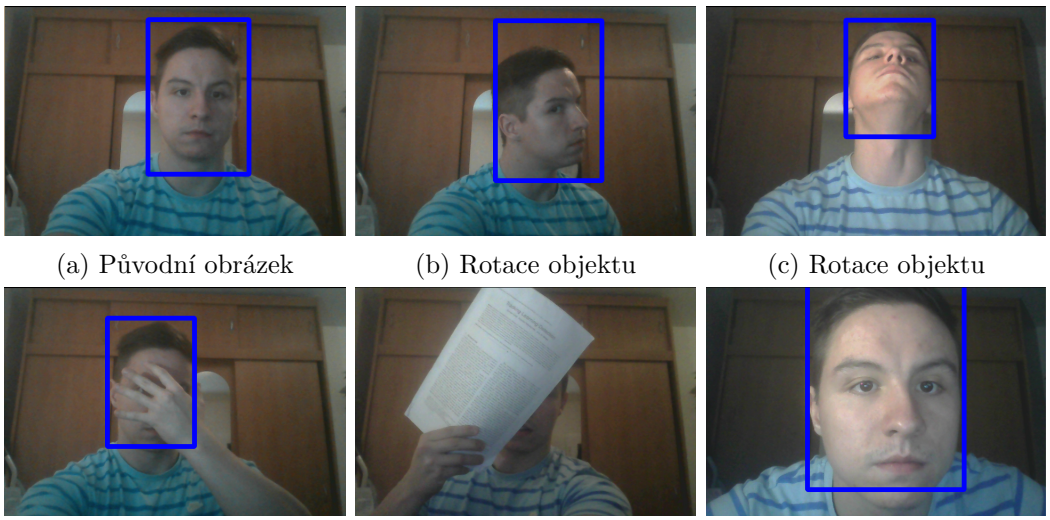
Obr. 4.7: Graf výkonnosti jednotlivých modelů.

[1] http://www.eecs.qmul.ac.uk/~andrea/avss2007_d.html

Při trénování modelu byly vyzkoušeny různé změny objektu, které jsou popsány v podkapitole 4.1. Na některých snímcích se objevují žluté obdelníky a modré, které značí s jakou jistotou se sleduje daný objekt. Z teorie je to tzv. posteriorní pravděpodobnost. Žluté obdelníky značí, že hodnota je menší než 0,7, modré potom značí hodnotu větší než 0,7. Obrázek 4.8 ukazuje tréninkový model obličeje. Tento model fungoval bez problémů. Na snímku 4.8a jde první snímek při započetí sledování objektu. Dále na snímcích 4.8b, 4.8c jde vidět rotace objektu a na 4.8d, 4.8e částečné zakrytí objektu. U snímku 4.8e bylo již zakrytí velké a tak bylo přerušeno sledování, které však po objevení celého objektu bylo opět spuštěno. Na snímku 4.8f pak jde vidět změna velikosti objektu.

Druhým natrénovaným modelem je dóza 4.9. U tohoto modelu nastávali situace, kdy se objevil žlutý obdelník. V takových případech je natrénováno málo snímků a musí se zacházet s dózou opatrně aby nebylo přerušeno sledování. Tyto žluté obdelníky jdou vidět na snímcích 4.9b, 4.9c, kdy docházelo k rotaci objektu a u snímku 4.9d, kdy objekt byl z velké části zakrytý. Na snímku 4.9e jde vidět přední strana obalu, ale na sledování objektu to nemělo vliv. Na posledním snímku 4.9f je vidět změna velikosti.

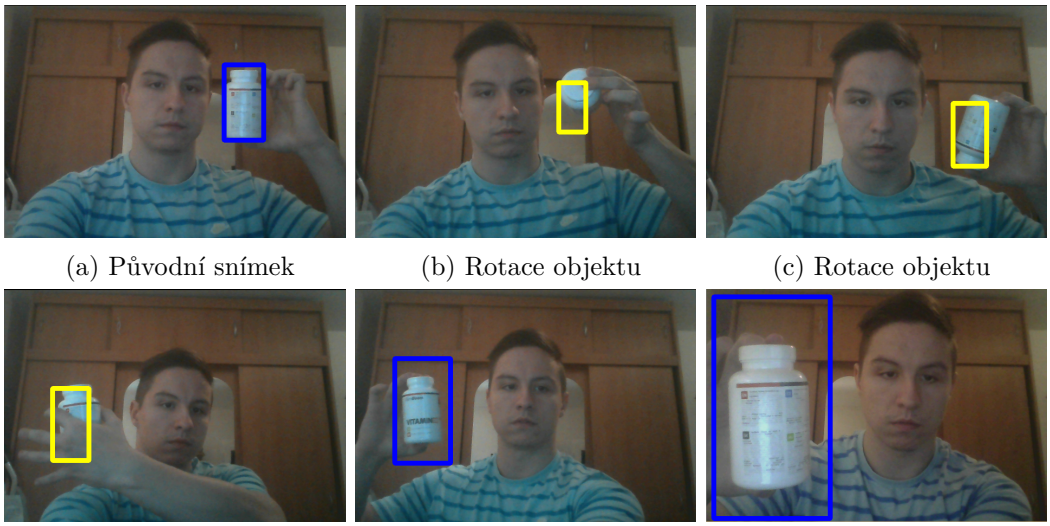
Posledním natrénovaným modelem je člověk z videosekvence 4.10. Označený sledovaný objekt jde vidět na snímku 4.10a. Poté se objekt pohyboval a ohraničující obdelník se mírně posunul, jak jde vidět na snímcích 4.10b a 4.10c. Na těchto snímcích jde i vidět žlutý ohraničující obdelník. Tento model byl nejvíce problémový, protože předtím, než sledovaný objekt odešel ze záběru, tak se ohraničující obdelník zasekl (obr. 4.10e) a nedokázal se obnovit i potom, co se člověk vrátil do záběru (obr. 4.10f).



(a) Původní obrázek (b) Rotace objektu (c) Rotace objektu

(d) Zakrytí objektu (e) Zakrytí objektu (f) Změna velikosti

Obr. 4.8: Natrénovaný model obličeje.



(a) Původní snímek (b) Rotace objektu (c) Rotace objektu

(d) Zakrytí objektu (e) Změna vzhledu (f) Změna velikosti

Obr. 4.9: Natrénovaný model dózy.

(a) Původní snímek (b) (c)

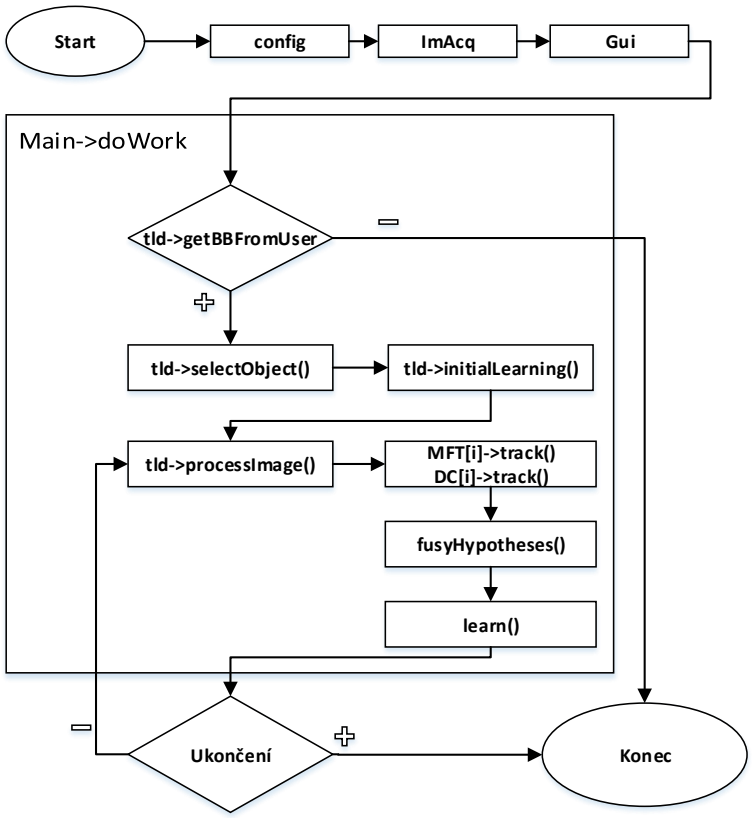
(d) (e) (f)

Obr. 4.10: Natrénovaný model člověka.

4.3 Sledování více objektů

4.3.1 Způsob fungování OpenTLD

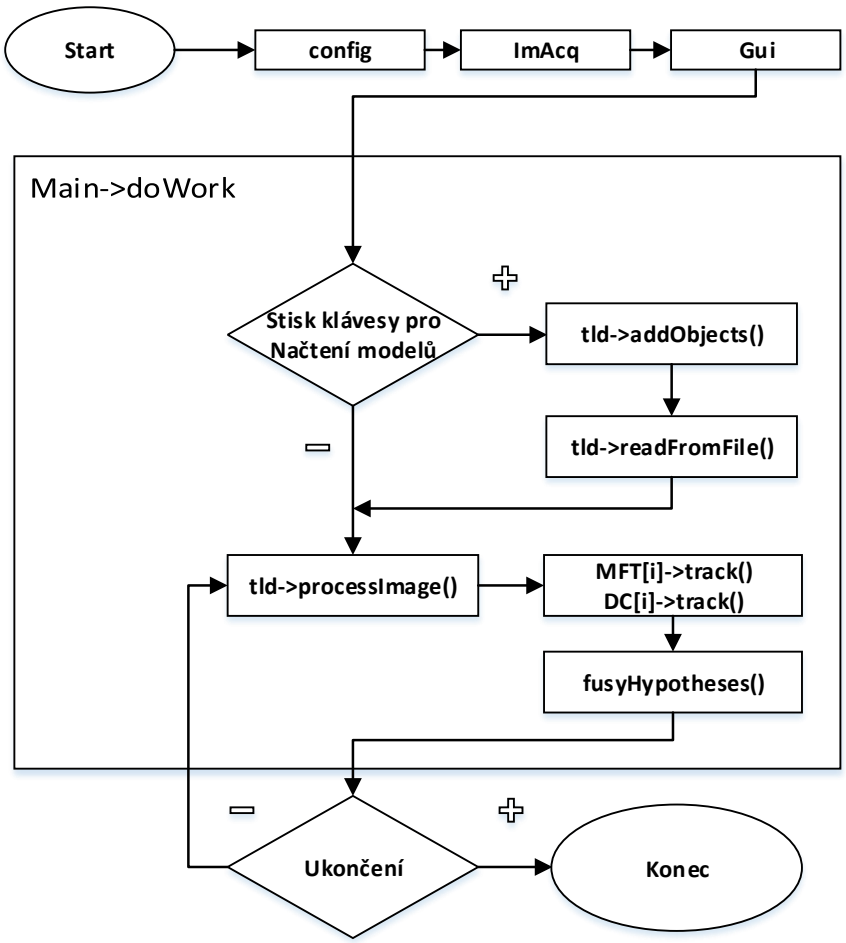
Hlavním rozšířením programu OpenTLD je schopnost sledovat a detekovat více objektů v reálném čase. V původní verzi je program schopen sledovat a trénovat pouze jeden objekt. V rozšířené verzi je trénován stále jen jeden objekt, ale program je schopen načíst více natrénovaných modelů a všechny tyto modely sledovat a detekovat zároveň. Trénovaní pouze jednoho modelu je výhodnější hlavně kvůli výkonnosti, která může být ovlivněna, pokud by bylo trénováno více objektů zároveň. Při sledování více objektů je vypnuto učení a tím je vylepšen výkon a počet snímků za sekundu, jelikož všechny modely jsou již předem natrénované. Pro lepší detekci a sledování je vhodné, aby byl objekt dostatečně natrénován. Pokud není objekt dostatečně natrénován, může docházet k chybné detekci nebo dokonce žádné. Na obr. 4.11 jde vidět, jakým způsobem funguje celý program a učení jednoho objektu. Pokud je vybrána špatná velikost objektu, tak je program s chybou ukončen. Pokud je vybrán objekt správné velikosti, tak program zahájí učení.



Obr. 4.11: Vývojový diagram učení jednoho objektu.

Uložení modelu probíhá pomocí stisknutí klávesy `e`. Model se pak uloží do stejné složky jako spustitelný soubor programu OpenTLD pod názvem model00. Další modely se pak jmenují obdobně, jen mají jiné číslo. Seznam všech klávesových zkratek je zmíněn v kapitole 4.5. Všechny uložené modely jsou v textovém formátu, ve kterém jsou údaje o daném modelu. Tyto údaje tvoří začátek modelu v x-ose a y-ose, jeho šířku a výšku.

Na obrázku 4.12 lze vidět, jak probíhá sledování více modelů zároveň. Načtení všech modelů se provádí stiskem klávesy `i`. Poté probíhá sledování a detekce bez učení. Modely jsou načteny postupně podle očíslování. Sledování a detekce pak probíhá u všech objektů postupně v jednom snímku a čeká se, až se zpracuje každý sledovaný model. Počet snímků za sekundu je pak ovlivněn počtem načtených modelů a jejich detekcí.



Obr. 4.12: Vývojový diagram sledování a detekce více objektů.

Výpis 4.1: Metoda processImage pro zpracování snímku

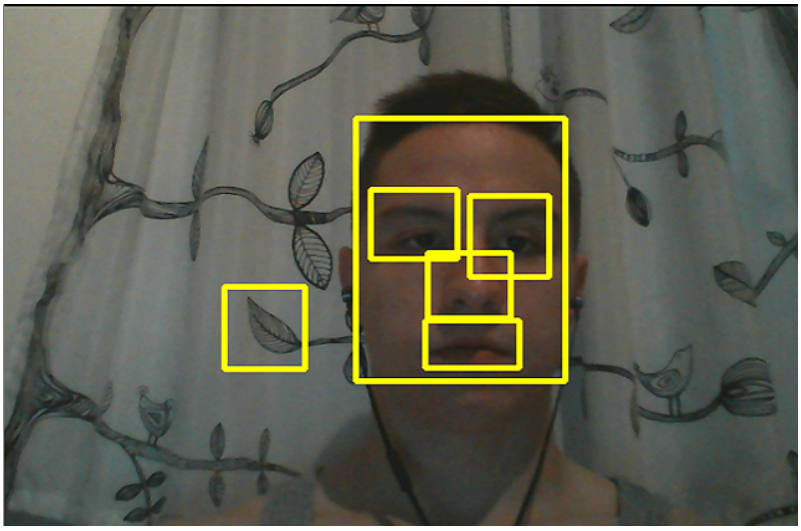
```
1  void TLD::processImage(constMat &img)
2  {
3    storeCurrentData();   //uloží předchozí data
4    Mat grey_frame;
5    cvtColor(img, grey_frame, CV_BGR2GRAY);
6    currImg = grey_frame; // načte další snímek
7
8    for (int i = 0; i < DC.size(); i++) // projde všechny modely
9      {
10     // pokud byl v~předchozím snímku model detekován
11     if (previousBB[i] != NULL)
12       {
13         // Sledování pomocí odhadu optického toku
14         MFT[i]->track(prevImg, currImg, previousBB[i]); ,
15             }
16         if (detectorEnabled && (!alternating ||
17           MFT[i]->trackerBB == NULL))
18             {
19             // Detekce pomocí kaskádové architektury
20               DC[i]->detect(grey_frame);
21             }
22          }
23     fuseHypotheses();
24     learn();  // učení
25 }
```

Ve výpisu 4.1 je vypsána metoda pro zpracování každého snímku. Před sledováním a detekcí je potřeba uložit hodnoty ze starého snímku a načíst nový snímek ke zpracování. Poté proběhne pro každý objekt sledování pomocí odhadu optického toku a detekce pomocí kaskádové architektury. V rozšíření jsou proměnné vytvořeny jako vector, který vytvoří dynamické pole a díky tomu je možné rozšířit počet objektů. Pro pokračování se čeká na detekci všech objektů ve snímku. Poté jsou v metodě `fuseHypotheses();` předány parametry ohraničujícím obdelníkům sledovače i detektoru, se kterými se dále pracuje a metoda `learn()` je použitá, v případě, že je povoleno učení. To je povoleno v případě sledování s učením jednoho objektu.

4.3.2 Konkrétní příklad

Pro měření přesnosti a počtu snímků za sekundu byl vybrán vstup z kamery o rozlišení 1280x960. Na videu jsou sledovány různé části na obličeji a objekt na pozadí. Při učení jednoho objektu byl počet snímků za sekundu v rozmezí 20-30 a video bylo většinu času plynulé. Při postupném natrénování šesti modelů byla sledována přesnost detekce všech modelů a počet výpadků ve videu, jak jde vidět v tabulce 4.1. Čím více modelů bylo sledováno, tím menší byl počet snímků za sekundu. Při sledování jednoho objektu bez učení se počet snímků za sekundu pohyboval okolo 40. Při sledování všech šesti modelů se počet snímků za sekundu pohyboval okolo 20.

Největší vliv na výkon a počet snímků za sekundu mají objekty, které nejsou detekovány. V takovém případě totiž musí kaskádová architektura prohlédnout všechny pod-okna. Pokles je tedy výrazně vyšší a podle počtu neodhalených modelů se počet snímků za sekundu může výrazně snížit. Pokud nebyl detekován jeden objekt, počet snímků za sekundu se pohyboval okolo 10, což je až o 50% méně než při detekci všech objektů. Pokud nebylo detekováno více objektů, počet snímků za sekundu klesl až ke 3. Nejlepšího výkonu a počtů snímků za sekundu je tedy dosaženo v případě, že všechny objekty jsou detekovány. Na největší přesnost detekce a sledování má také velký vliv natrénovanost objektů. Pokud jsou objekty dostatečně natrénovány, dochází u jejich detekce k méně častým výpadkům a tím pádem k lepšímu výkonu OpenTLD.



Obr. 4.13: Snímek z videa.

Na videu [27] lze vidět, že některé modely nejsou dostatečně natrénovány a detektor není schopen detekovat objekty po určitých změnách objektu. Video slouží

k demonstraci fungování sledování a detekce všech načtených modelů. Všechny modely byly natrénovány na vzorku 300 snímků, kde byly různé změny, jako například rotace objektu nebo vzdálenost od webkamery. Tento počet snímků nemusí být v některých případech dostatečný a je potřeba natrénovat model na více snímcích. Nedostatečná natrénovanost by mohla dělat problém např. u lidí, kde se jejich podoba mění často a rychle. Poté byly načteny všechny modely a otestovány různé situace jako překrytí, rotace objektu, změna velikosti i změna podoby.

Přesnost byla vypočítána podle vzorce:

$$P_d = \frac{I_t - I_v}{0,01 \cdot I_t}, \tag{4.1}$$

kde P_d je přesnost detekce, I_t je celkový počet testovacích snímků a I_v je počet snímku, ve kterých došlo k výpadku. 0,01 je dosazeno kvůli tomu, aby výsledek vycházel v procentech

Tab. 4.1: Přesnost sledování jednotlivých objektů při učení v OpenTLD

Objekty	**Výpadky**	**Přesnost** [%]
List na zácloně	27	91
Levé oko	14	95,34
Pravé oko	1	99,67
Nos	6	98
Pusa	1	99,67
Obličej	4	98,67

4.4 Paralelizace, testování videí

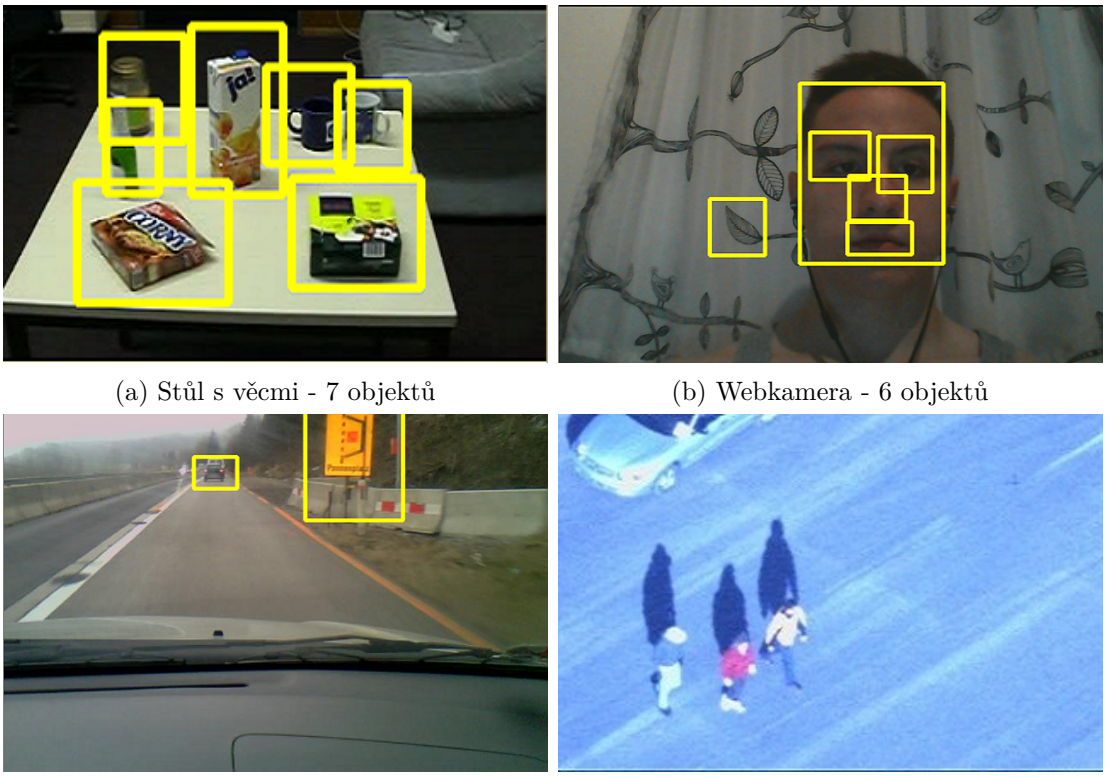
4.4.1 Paralelizace

Jedna z důležitých věcí, která je schopna pozitivně ovlivnit výkon a počet snímků za sekundu je paralelizace. Díky paralelizaci je možnost využít plný potenciál procesoru a použít všechny jeho vlákna. Tím je proces urychlen. V jazyku C++ lze k paralelizaci využít **OpenMP**. K jeho implementaci je potřeba využít kompilátor, který OpenMP obsahuje (v této práci ho využívá Visual Studio). OpenMP je jednoduše implementovatelný a může paralelizovat pouze kritické části programu. Jeho využití je však omezeno pouze na počítače se sdílenou pamětí. Paralelizace byla v tomto případě aplikována pouze na kaskádovou architekturu (detekci objektu), protože je výpočetně náročnější než sledování pomocí odhadu optického toku. Zároveň kaskádová architektura prochází několik tisíc pod-oken, kdežto odhad optického

toku porovnává pouze předešlí a aktuální snímek. Aplikování paralelizace na učení bylo v tomto případě neúspěšné a tím pádem počet snímků za sekundu je menší při učení objektu.

4.4.2 Testování paralelizace a přesnosti detekce

V následujících příkladech byla porovnána paralelizace pro čtyři různé vstupy s různým počtem modelů. Dále byl simulován vliv detekce objektů na počet snímků za sekundu. Nejlepší výkon byl v případě detekovatelnosti všech objektů s paralelizací. Nejhorších výsledků bylo dosaženo v případě, že některé objekty nebyly detekovány a nebyla povolena paralelizace. Byly porovnány čtyři různé vstupy s různou náročností detekce, u kterých bylo dosaženo různých výsledků. Všechny testy proběhli na notebooku s parametry uvedenými v kapitole 4.5.



(a) Stůl s věcmi - 7 objektů

(b) Webkamera - 6 objektů

(c) Auta - 4 objekty

(d) Lidi - 3 objekty

Obr. 4.14: Testované modely pro sledování více objektů

Porovnány byly dvě situace. 1. situace (označena jako A) byla taková, že všechny sledované modely byly zobrazeny ve snímku. V 2. situaci (označena jako B) byly některé modely nějakým způsobem zakryty. V této situaci se počet snímků za sekundu lišil podle toho, kolik modelů bylo zakryto. Počet snímků za sekundu v situaci B byl mnohem menší než v A ,protože detektor musí projít všechny pod-okna, kdežto

v první situaci musí projít jen tolik pod-oken, než objeví hledaný objekt. Způsob trénování u webkamery je popsán v kapitole 4.3.2. U videí (4.14a, 4.14c, 4.14d) byly objekty trénovány v první polovině videa. Poté bylo video znovu spuštěno, modely byly naimportovány a sledování objektů probíhalo po celou dobu videa. Trénovací a testovací data byla podobná pro tyto videa. V případě webkamery se mírně lišily a byly testovány nové situace. Nejlepších výsledků bylo dosaženo pro video, kde jsou věci položené na stole. U tohoto videa proběhla detekce bez výpadků. Podobných vysledků bylo dosaženo ve videu z webkamery, kde detekce probíhala taktéž bez výpadků. Horší výsledky se vyskytly u videa s auty, kde počet snímků za sekundu byl nižší a detekce objektů byla v chvílemi nepřesná, avšak objekty byly detekovány. Největší problém byl ve videu s lidmi, kde byl problém při detekcí od začátku a objekty bylo obtížné i natrénovat. Video s lidmi je uvedeno jako nevhodný příklad pro sledování a detekci pomocí TLD. V tabulce 4.2 je vidět počet snímků za sekundu při sledování všech objektů. V této tabulce jsou pro video s auty (4.14c) jsou prázdná místa v situaci A, protože nenastal případ, kde by všechny sledované objekty byl v záběru současně. V tabulce 4.3 je pak vidět počet snímků při učení jednoho objektu. Snímky čekají, až se provedou všechny detekce v obraze, až poté se přejde na další snímek. Tímto způsobem je také měřen počet snímků za sekundu. Hodnoty v tabulkách jsou převedeny i do grafů 4.15 a 4.16

Tab. 4.2: Počet snímků za vteřinu při sledování všech objektů.

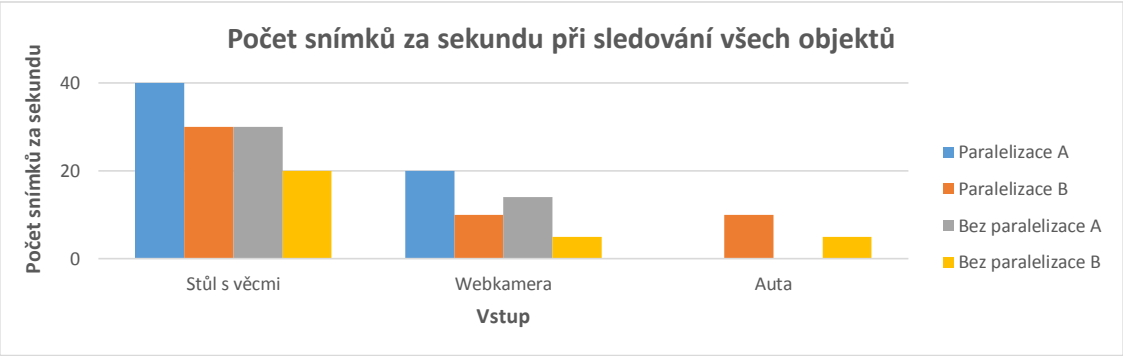
			Počet snímků za sekundu [FPS]			
			Paralelizace		**Bez paralelizace**	
Název videa	**Počet objektů**	**Rozlišení**	A	B	A	B
Stůl s věcmi	7	320x240	40	30	30	20
Webkamera	6	1280x960	20	10	14	5
Auta	4	640x480		10		5

Tab. 4.3: Počet snímků za vteřinu při učení jednoho objektu.

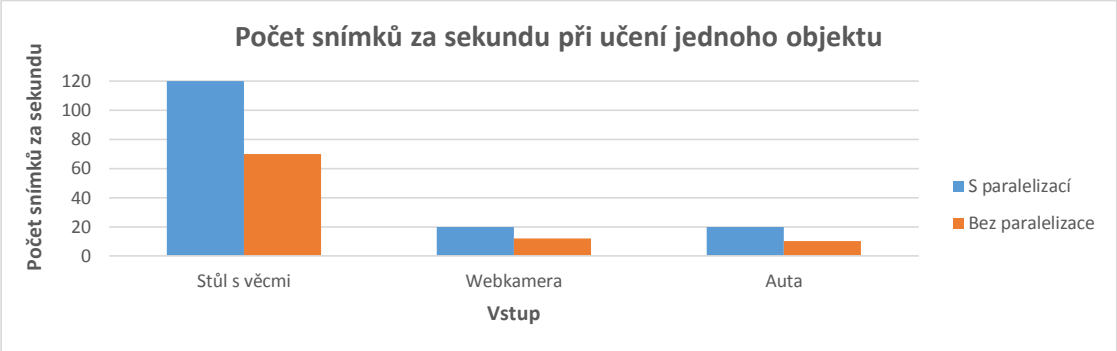
			Počet snímků za sekundu [FPS]	
Název videa	**Počet objektů**	**Rozlišení**	**Paralelizace**	**Bez paralelizace**
Stůl s věcmi	7	320x240	120	70
Webkamera	6	1280x960	20	12
Auta	4	640x480	20	10

Z tabulek lze vyčíst, že v některých případech je paralelizace schopna zlepšit výkon o 50%. Bez paralelizace bylo využito pouze jedno vlákno a CPU bylo zatíženo maximálně na 25%. S paralelizací byly využity všechny vlákna a využití CPU se pohybovalo okolo 90%. I přesto, že se využívá CPU 4x více, nebyla výpočetní rychlost 4x lepší.

Díky těmto výsledkům se dá vyvodit, že TLD se nehodí pro objekty, které nejsou dostatečně naučeny a mění svůj tvar příliš rychle. Jelikož nevyužívá žádné šablony, TLD se spolehá pouze na dostatečné naučení objektu.



Obr. 4.15: Počet snímků za sekundu při sledování všech objektů.



Obr. 4.16: Počet snímků za vteřinu při učení jednoho objektu.

4.5 Klávesové zkratky a testované PC

Program je ovládán pomocí kláves na klávesnici. Notebook, na kterém probíhalo testování byl Lenovo B590.
Seznam klávesových zkratek:
H - Nápověda v konzolovém okně.
Q - Vypnutí programu OpenTLD.
A - Mód přepínání (při zapnutí se vypne detektor, pokud je sledovač dostupný).
E - Exportování jednoho modelu do složky.
I - Importování všech modelů ze složky.
R - Výběr nového modelu k učení.
C - Přerušení sledování/detekce/učení.

Tab. 4.4: Parametry PC, na kterém bylo OpenTLD testováno

Operační systém	Windows 8
Typ procesoru	Intel Core i3 3110M Ivy Bridge
Počet jader/vláken procesoru	2 / 4
Frekvence procesoru	2,4 GHz (2 400 MHz)
Velikost operační paměti	4 GB / DDR3 / 1 600 MHz (1,6 GHz)
Grafická karta	Intel HD Graphics 4000
HDD	500 GB / 5400 ot./min

5 ZÁVĚR

Tato práce se zabývala online trénováním detektoru pro sledování objektů. Cílem bylo zprovoznit tento detektor a seznámit se s problematikou učení detektoru a sledování objektu. Dále byl natrénovány nové modely a rozšíření programu o sledování více objektů zároveň s aplikováním paralelizace. V 1. kapitole bylo popsáno, kam směřuje v dnešní době problematika sledování objektu. V 2. kapitole byl představen TLD framework, který se skládá ze tří hlavních částí: sledování objektu, detekce objektu a učení detektoru. Sledovací část je zajištěna pomocí tzv. rekurzivního sledování, detekce objektu funguje na bázi kaskádové architektury a učení detektoru pomocí P-N učení. V 3. kapitole je potom návrh řešení a postup instalace pro zprovoznění OpenTLD. V kapitole 4 se řešily jednotlivé problémy vyskytující se u sledovače a byly představeny natrénované modely. Dále je v kapitole řešena aplikace paralelizace a sledování více objektů.

V této práci bylo zprovozněno OpenTLD, které slouží ke sledování objektu. Aby OpenTLD správně fungovalo, je potřeba zajistit jeho detektorovou část. Toho je dosaženo díky knihovnám OpenCV. Díky programu CMake je umožněno tyto dvě části spojit ve funkční celek. Ve Visual Studiu je potom projekt sestaven a vytvořen spouštěcí soubor. Dále je v práci vytvořen postup instalace a popsání problémů s kterými se TLD potýká a jak zvládá změny sledovaného objektu. V první části byly natrénovaný jednotlivé modely z webkamery a videa a byla porovnána přesnost sledování a jejich výkon.

Hlavním přínosem práce bylo rozšíření OpenTLD o paralelní detekci a sledování více objektů. Paralelizace je dosažena pomocí OpenMP. V práci je pak porovnáno, jaký vliv na výkon má procesor s využitím všech vláken. Porovnání je uvedeno na dvou rozdílných vstupech, které se liší rozlišením. Na několika vstupech pak byla ověřena přesnost detekce a výkon s paralelizací a bez paralelizace. Nejlepších výsledků bylo dosaženo při paralelizaci, kdy všechny objekty byly detekovány. Nejlepší schopnost detekce měly statické objekty, kde se pouze měnila poloha kamery. Naopak nejhorších výsledků bylo dosaženo při málo naučených objektech, které rychle měnili svůj tvar.

Návrh na budoucí rozšíření této práce je možné vést více směry. Jednou z možností je vytvoření aplikace na mobilní telefon nebo program aplikovat na nějaký vhodné sledovací zařízení. Další možností je vylepšit proces sledování a aplikovat novější metody, které by měly pozitivní dopad na výkon.

LITERATURA

[1] NEBEHAY, G. *Robust Object Tracking Based on Tracking-Learning-Detection* [online], Faculty of Informatics, TU Vienna, 2012. Dostupné na internetu: <http://www.caa.tuwien.ac.at/cvl/wp-content/uploads/2015/12/thesis.pdf>

[2] BRADSKI, G. R.; KEHLER, A. *Learning OpenCV* Sebastopol: O'Reilly, 2008, xvii, 555 s. : ill. ISBN 978-0-596-51613-0.

[3] BRUCE D. LUCAS, KANADE, T. *An Iterative Image Registration Technique with an Application to Stereo Vision* Proceedings of the 7th International Joint Conference on Artificial Intelligence, 1981, s. 674-679. Dostupné na internetu: <https://cseweb.ucsd.edu/classes/sp02/cse252/lucaskanade81.pdf>

[4] KYME, A.; SE, S.; MEIKLE, S.; ANGELIS, G.; RYDER, W.; POPOVIC, K.; YATIGAMMANA, D.; FULTON, R. *Markerless Motion Tracking of Awake Animals in Positron Emission Tomography* [online], IEEE Transactions on Medical Imaging, 2014, USA: IEEE, 1411, 33(11), 2180-2190. DOI: 10.1109/TMI.2014.2332821. ISSN 0278-0062.

[5] RAMAKRISHNA, V.; SHEIKH, Y.; KANADE, T. *Tracking Human Pose by Tracking Symmetric Parts* [online], Computer Vision and Pattern Recognition (CVPR), 2013, IEEE, 1306, s. 3728-3735. DOI: 10.1109/CVPR.2013.478. ISSN 1063-6919.

[6] PARAVATI, G.; ESPOSITO, S. *Relevance-Based Template Matching for Tracking Targets in FLIR Imagery* [online], Sensors, 2014, 14(8), 14106. DOI: 10.3390/s140814106.

[7] SELKA, F.; NICOLAU, S.; AGNUS, V.; BESSAID, A.; MARESCAUX, J.; SOLER, L. *Context-specific selection of algorithms for recursive feature tracking in endoscopic image using a new methodology* [online], Computerized Medical Imaging and Graphics, 2014, Elsevier, 1503, 40, 49-61. DOI: 10.1016/j.compmedimag.2014.11.012. ISSN 0895-6111.

[8] HOLZER, S.; ILIC, S.; TAN, D.; POLLEFEYS, M.; NAVAB, N.; *Efficient Learning of Linear Predictors for Template Tracking* [online], International Journal of Computer Vision, 2015, Boston: Springer US, 1501, 111(1), 12-28 . DOI: 10.1007/s11263-014-0729-1. ISSN 0920-5691.

[9] BERG, A. *Detection and Tracking in Thermal Infrared Imagery* [online], Linköping University, Department of Electrical Enginnering, 2016, DOI: 10.3384/lic.diva-126955. ISBN 978-91-7685-789-2 . ISSN 0280-7971.

[10] YANA, D.; POPOVIC, J.; RUNBORG, O.; *An Adaptive Fast Interface Tracking Method* [online], Journal of Computational Mathematics, 2015, 1511, 33(6), 576-586. DOI: 10.4208/jcm.1503-m4532. ISSN 02549409.

[11] ASHA, C.S.; NARASIMHADHAN, A.V. *Adaptive Learning Rate for Visual Tracking Using Correlation Filters* [online], Procedia Computer Science, Elsevier B.V, 2016, 89, 614-622. DOI: 10.1016/j.procs.2016.06.023. ISSN 1877-0509.

[12] WEI, S.; KOSOROK, M.R.. *Latent Supervised Learning* [online], Journal of the American Statistical Association, 2013, 1309, 108(503), 957-970. DOI: 10.1080/01621459.2013.789695. ISSN 0162-1459.

[13] ZHU, S.; SUN, X.; JIN, D.; *Multi-view semi-supervised learning for image classification* [online], Neurocomputing, Elsevier B.V, 2016, 208, 136-142. DOI: 10.1016/j.neucom.2016.02.072. ISSN 0925-2312.

[14] LEBEDA, K.; HADFIELD, S.; MATAS, J.; BOWDEN, R. *Texture-Independent Long-Term Tracking Using Virtual Corners* [online], IEEE Transactions on Image processing, 2015, USA: IEEE, 1601, 25(1), 359-371. DOI: 10.1109/TIP.2015.2497141. ISSN 1057-7149.

[15] QINGJI, G.; ZHENG, C.Z.; DANDAN, H. *Long-term tracking method on ground moving target of UAV* [online], Guidance, Navigation and Control Conference (CGNCC), 2014, IEEE, 1408, , 2429-2432. DOI: 10.1109/CGNCC.2014.7007550. ISBN 978-1-4799-4700-3.

[16] KALAL, Z.; MIKOLAJCZYK, K.; MATAS, J. *Tracking-Learning-Detection* [online], IEEE, Transactions on Pattern Analysis and Machine Learning, 2012, USA: IEEE, 1207, 34(7), 1409-1422. DOI: 10.1109/TPAMI.2011.239. ISSN 0162-8828.

[17] VIOLA, P.; JONES, M. *Rapid object detection using a boosted cascade of simple features* [online], Computer Vision and Pattern Recognition, USA: IEEE, 2001, 1, I-I. DOI: 10.1109/CVPR.2001.990517. ISBN 0-7695-1272-0. ISSN 1063-6919.

[18] ROSENBAUM, D.; WEISS, Y. *Beyond Brightness Constancy: Learning Noise Models for Optical Flow* [online], School of Computer Science and Engineering Hebrew University of Jerusalem, 2016 Dostupné z internetu: <https://arxiv.org/pdf/1604.02815v1.pdf>

[19] KALAL, Z.; MIKOLAJCZYK, K.; MATAS, J.; *Forward-Backward Error: Automatic Detection of Tracking Failures* [online], International Conference

on Pattern Recognition, 2010, IEEE Publishing, 1008, s. 2756-2759. DOI: 10.1109/ICPR.2010.675. ISBN 978-1-4244-7542-1. ISSN 1051-4651.

[20] MATSUKAWA, T.; KUIRITA, T. *Image representation for generic object recognition using higher-order local autocorrelation features on posterior probability images* [online] Pattern Recognition, 2012, 1202, 45(2), 707-719. DOI: 10.1016/j.patcog.2011.07.018. ISSN 00313203.

[21] FANG, X.; LU, Y; LI, Z.; YU, L.; CHEN, Y. *Kernel representation-based nearest neighbor classifier* [online], Optik - International Journal for Light and Electron Optics, 2014, Elsevier, 1405, 125(10), 2320-2326. DOI: 10.1016/j.ijleo.2013.10.074. ISSN 0030-4026.

[22] KAWAKITA, M.; TAKEUCHI, J. *Safe semi-supervised learning based on weighted likelihood* [online], Neural Networks, 2014, Elsevier, 1405, 53, 146-164. DOI: 10.1016/j.neunet.2014.01.016. ISSN 0893-6080.

[23] FU, C.; YANG. Y *Low density separation as a stopping criterion for active learning SVM* [online], Intelligent Data Analysis, 2015, 19(4), 727-741. DOI: 10.3233/IDA-150742. ISSN 1088467X.

[24] ROSENBERG, C.; HEBERT, M.; SCHNEIDERMAN, H.; *Semi-Supervised Self-Training of Object Detection Models* [online], Application of Computer Vision, 2005, IEEE, 0501, 1, s. 29-36. DOI: 10.1109/ACVMOT.2005.107. ISBN 0-7695-2271-8.

[25] DARNSTDT, M.; SIMON, H. U., SZÖRÉNYI, B.; *Supervised learning and Co-training* [online], Theoretical Computer Science, Elsevier B.V, 2014, 519, 68-87. DOI: 10.1016/j.tcs.2013.09.020. ISSN 0304-3975.

[26] KALAL, Z.; MATAS, J.; MIKOLAJCZYK, K.; *P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints* [online], IEEE Conference on Computer Vision and Pattern Recognition, 2010, IEEE Publishing, 1006, s. 49-56. DOI: 10.1109/CVPR.2010.5540231. ISBN 978-1-4244-6984-0. ISSN 1063-6919.

[27] PRIBYL, J.; *Multi object tracking OpenTLD.* [online], Youtube [cit. 2017-06-01]. Dostupné z internetu: <https://www.youtube.com/watch?v=Ss81kokWGPc>

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

BSD	Berkeley Software Distribution – Softwarová distribuce univerzity v Berkeley
CPU	Central Processor Unit – Centrální procesorová jednotka
d	Vektor přemístění
EM	Expectation-Maximization – Očekávání maximalizace
FPS	Prames per second – Snímky za sekundu
h	Výška pod-okna
k-NN	Nearest neighbour classifier – klasifikátor nejbližších sousedů
I	Snímek
I'	Gradient snímku I
J	Další snímek
LK	Lucasova a Kanadeho metoda
NCC	Normalised Corellation Coefficient – Normalizovaný korelační koeficient
p	Podobnost obklopující oblasti pixelu
P_i	Posteriorní pravděpodobnost
P_n	Směrodatná odchylka
QVGA	Quarter Video Graphics Array – video o rozměrech 320x240
R	Velikost množiny všech pod-oken
S^c	Konzervativní pravděpodobnost
s	Kroková změna
TLD	Tracking-Learning-Detection – Sledování-učení-detekce
UI	User Interface – Uživatelské rozhraní
w	Šířka pod-okna
W	Oblast okolo pixelu

6 OBSAH PŘILOŽENÉHO CD

K bakalářské práci je přiloženo médium ve formě CD, na kterém se nachází elektronická verze bakalářské práce. Dále se zde nachází zdrojové kódy programu, které jsou rozšířeny o problematiku popsanou v práci ve vývojovém prostředí Visual studio a také samotný program. Zdrojové kódy se nachází ve složce `src` a spustitelný soubor ve složce `opentld`. CD je umístěno na zadní části desky.